Network Design with node constraints and Maximum Network Lifetime problems

Lee, Sang Hyuk

Awarding institution:
King's College London

Network Design with node constraints and Maximum Network Lifetime problems

Sang Hyuk Lee

Department of Informatics

King's College London

A thesis submitted for the degree of

Doctor of Philosophy

October 2016

Dedicate to my parents Won-woo Lee and Mi-kyung Park.

Acknowledgements

First and foremost, I am deeply indebted to my supervisor, Prof. Tomasz Radzik, for his invaluable guidance and endless support during my PhD study. I have been extremely lucky to have him as my supervisor. He cared so much about my work and always gave fruitful advises. His guidance and patience were paramount in the formulation of this thesis. Without him, this would not have been possible.

I would also like to thank Prof. Colin Cooper for giving me an opportunity to participate in SAMSUNG project. It was a great honour to work with him and he helped me to broaden my knowledge in many other areas. I also thank him for providing the funding and for giving me the opportunity to attend the conference.

I have been very fortunate to be a part of Algorithms & Bioinformatics Group (ABG) here at King's College London. I would like to thank all the past and present members of the ABG. In particular, I thank Prof. Maxime Crochemore, Prof. Costas Iliopoulos, Dr Sophia Tsoka, Dr Kathleen Steinhofel, and Dr Solon Pissis and lab mates Yiannis, Michalis, Ouala, and Adulrahman.

The most encouragement and support that I received in completing this thesis has come from my wife Erica. Words cannot express the debt of gratitude I owe to her for the love and support she gave to me. Suffice it to say that if it were not for her sacrifice, I would not be where I am now.

Last but not the least, I would like to thank my parents and my sister for always believing in me, always supporting me, and always helping me to achieve my dream. My hard-working

parents have sacrificed their lives for me and provided unconditional love and care. The love they gave to me will live in my heart.

Abstract

In the thesis, we consider the *Directed Weighted Degree Constrained Network Design* (DWDCN) problem and its applications to *Maximum Network Lifetime* (MNL) problems in wireless ad-hoc networks.

The goal of the DWDCN problem is to find a minimum-cost subgraph satisfying the specified *connectivity requirements* and the specified *degree bounds*. This problem has many variants, depending on the type of the connectivity requirements and on the type of the degree bounds. We consider a general case when the connectivity requirements are defined by an *intersecting* or *crossing* supermodular set function and the degree bounds are defined for the out-degrees of nodes or for the in-degrees or both. Since most of the DWDCN problems are known to be NP-hard, we consider approximation algorithms. While requiring that all connectivity constraints are (strictly) satisfied, we allow approximation of both the total cost and the degree bounds. More specifically, an (α,β) bi-criteria approximation algorithm for an DWDCN problem computes in polynomial time a subgraph which satisfies the connectivity requirements but may violate the optimality of the cost by a factor α and degree bounds by a factor β. We improve a number of previous (α,β)-approximation bounds, for example, we show a $(2,5)$-approximation bound for the DWDCN problem with out-degree constraints and connectivity requirements defined by an intersecting supermodular function. The previous best bounds were $(2,7)$ and $(3,6)$-approximations.

One application of the DWDCN algorithm is to solve the MNL problem. In an MNL problem, we are given a wireless ad-hoc network with an edge-weight function representing

the energy costs of individual transmissions, and a node function representing the initial energy of nodes. The communication tasks we consider are *unicast*, *broadcast*, *convergecast* and *mixedcast*. The goal is to compute a schedule of individual transmissions to perform a specified communication task as many times as possible before the energy of the nodes is depleted. Using our approximation bounds for DWDCN problems, we improve the previous approximation algorithms for MNL problems. For example, we show a polynomial time algorithm which computes a schedule allowing $\lfloor k_{opt}/5 \rfloor$ rounds of broadcasting, where k_{opt} is the optimal number of rounds. This improves the previous best approximation bound of $\lfloor k_{opt}/36 \rfloor$.

We also conduct experimental evaluation of the considered MNL approximation algorithms, comparing the quality of the computed solutions with upper bounds and with solutions obtained by heuristics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Network design problems with weighted degree constraints

Network design problems form one of the main topics in combinatorial optimisation, approximation algorithms, and operations research. In a typical instance of a network deign problem, we are given a graph $G = (V, E)$, non-negative edge-costs $c(e)$ for all $e \in E$, and connectivity requirements. The objective is to find a minimum cost subgraph H of G, which satisfies the specified connectivity requirements. Examples of such problems include a wide variety of classical problems such as the *minimum spanning tree problem*, the *shortest path problem*, and the *travelling salesman problem*.

In a more general class of *degree-bounded network design* problems in addition to connectivity requirements, we are also given *degree constraints* at nodes. Constraints of this type arise naturally in various practical applications in domains such as communication networks, vehicle routing, and VLSI chip design [1, 59, 3, 56]. Degree constraints are used to model limits on node's *resources* or admissible *workload*. The objective of this type of network design problems is to find a minimum cost subgraph which satisfies the specified connectivity requirements as well as the degree constraints (bounds) on the nodes. A well known example is the *minimum*

bounded degree spanning tree problem [59, 23, 62]. A number of other degree-bounded network design problems have been considered, for example in [1, 38, 35, 14, 37, 46]. Adding degree constraints may increase considerably the computational complexity of a network design problem. For example, the minimum spanning tree can be solved easily in polynomial time, however, if we have degree constraints the problem becomes NP-hard [21].

In this thesis, we consider degree-bounded network design problems for *directed weighted graphs* with *weighted degree bounds* at nodes. Such problems are referred to as *Directed Weighted Degree Constrained Network Design* (DWDCN) *problems* [52]. The goal of a DWDCN problem is to find a minimum cost subgraph, which satisfies the specified *connectivity requirements* and the *weighted degree constraints*. DWDCN problems have many variants, depending on the type of the connectivity requirements and on the type of the degree bounds. In this thesis, we consider a general case when the connectivity requirements are defined by an *intersecting* or *crossing* supermodular set function and the degree bounds are defined for the *weighted out-degrees* of nodes or the *weighted in-degrees* or *both*.

1.2 Maximum network lifetime problems for wireless ad-hoc networks

Unicast, *broadcast* and *convergecast* are the fundamental communication tasks in wireless ad-hoc networks. Unicast is one-to-one communication, where information held in one node, called the *source*, is transmitted to another node, called the *destination*, possibly via intermediate nodes. Broadcast is one-to-all communication, where information held in one source node is transmitted to all other nodes. Convergecast can be viewed as the opposite to broadcast: information held in every node is transmitted to one specified node, called the sink or the destination. Many network operations and services such as information dissemination and data collection rely on these three communication tasks.

The nodes of a wireless ad-hoc network are often battery-powered, but are intended to operate over a long period of time. Typical applications for such networks include environmental monitoring and military surveillance, where replacing or recharging the batteries may not be easy, or even not possible at all. Therefore, an important design objective for communication algorithms is to optimise the energy efficiency, so that the *network lifetime* is maximised. When the battery of a node is depleted, then the communication protocol has to be adjusted and in the worst case scenario the network may no longer operate (may become disconnected).

A wide range of optimization problems modelling the energy efficiency in ad-hoc wireless networks have been proposed. One general approach is to focus on a *single session*, aiming to minimize the energy used to complete one specified communication task [43, 69, 71]. The example of this approach is the Minimum Energy Broadcast problem [8, 68, 69, 71] with the objective of finding a broadcast tree which minimises the total energy cost. The other general approach is to consider *multiple sessions* with the aim of maximising the *lifetime* of the network, which could mean, for example, maximising the number of times that specified communication tasks can be repeated until the first node depletes all its energy [13, 51, 54, 57, 58, 61]. This approach is typically employed in continuous monitoring applications, where periodic data gathering (convergecast) or reporting (unicast) have to be performed. For such applications, the first, "greedy" approach of optimizing only the current session may give sub-optimal solutions. This is because the network lifetime does not solely depend on the energy spent while performing a specified communication task, but also on the remaining battery capacity of the individual nodes.

In this thesis we follow the second, "global" approach to energy efficient communication, and consider a class of *Maximum Network Lifetime* (MNL) problems [13, 54, 55, 57]. A problem of this class is given by a specification of a network (node-to-node connections, communication costs, initial capacities of node batteries, etc.) and a specification of a communication task (e.g., a broadcast from a given node). This communication task is to be executed periodically, as many times as possible. We refer to one execution of this task as one communication round. The output is a collection of *routing topologies* such that each routing

topology defines one execution of the specified communication task (one round). The objective is to maximise the number of communication rounds, that is, to maximise the network lifetime. The constraints are that every node must have sufficient battery capacity to participate in all rounds.

The communication tasks which we consider for the MNL problems are broadcast, convergecast, and unicast. In addition to these basic communication tasks, we also consider so-called *mixedcast*, which is a combination of these three tasks. The MNL mixedcast problem was introduced in [54] as a problem of designing the maximum number of communication rounds such that each round consists of τ broadcasts and γ convergecasts, where τ and γ are given non-negative integers. We will follow this definition, but our method can be also applied to a *generalized mixedcast*, when all three types of communication tasks can be combined, and more than one task of each type can be specified. For example, we might require that each round consists of two broadcasts from each of the given source nodes $r_1, r_2, \ldots, r_q$ and one convergecast to the destination node r_0.

1.3 Summary of the contribution of the thesis

To state the contribution of this thesis, we have to introduce some necessary definitions. (These definitions will be repeated and expanded with further background in Chapter 2). For given two sets $X, Y \subseteq V$, X and Y *intersect* if $X \cap Y \neq \emptyset$ and *cross* if all sets $X \cap Y$, $X \setminus Y$, $Y \setminus X$, $V \setminus (X \cup Y)$ are non-empty. A set function f on V is *intersecting supermodular* (resp. *crossing supermodular*) if any sets $X, Y \subseteq V$ that intersect (resp. cross) satisfy the condition of supermodularity:

$$f(X) + f(Y) \leq f(X \cap Y) + f(X \cup Y).$$

Each intersecting supermodular set function f is also crossing supermodular set function. Given a directed graph $G = (V, E)$ and a non-negative integral intersecting (or crossing) supermodular set function f on V, a subgraph $H = (V, F)$ of G is *f-connected* or satisfies the connectivity

requirement f, if there are at least $f(S)$ incoming edges to every subset $S \subseteq V$. For example, the connectivity requirements that for each $v \in V$, there is a path in H from (given) r to v, can be specified by the intersecting supermodular function f such that $f(S) = 1$ for all $\emptyset \neq S \subseteq V \setminus \{r\}$, and $f(S) = 0$ otherwise. The connectivity requirements that for every pair of nodes $u, v \in V$, there are at least k directed edge-disjoint paths in H from u to v, can be specified by the crossing supermodular function f such that $f(S) = k$ for all $\emptyset \neq S \subsetneq V$ and $f(\emptyset) = f(V) = 0$.

Table 1.1 Approximation bounds of polynomial-time algorithms for the DWDCN problems with weighted out-degree or weighted in-degree or both weighted out- and in-degree constraints and intersecting or crossing supermodular connectivity requirements. The shown previous results are due to Nutov [52].

Previous results		
Weighted degree constraints	intersecting supermodular f	crossing supermodular f
out-degree	$(2,7)$ $(3,6)$	$(3,(7+\min\{4,f_{\max}\}))$ $(4,(6+\min\{4,f_{\max}\}))$
in-degree	$(1,\min\{4,f_{\max}\})$	$(3,(7+\min\{4,f_{\max}\}))$ $(4,(6+\min\{4,f_{\max}\}))$
both out- and in-degree	$(2,7,\min\{6,f_{\max}\})$ $(3,6,\min\{8,f_{\max}\})$	$(4,(7+\min\{6,f_{\max}\}),(7+\min\{6,f_{\max}\}))$ $(6,(6+\min\{8,f_{\max}\}),(6+\min\{8,f_{\max}\}))$
Our results		
	intersecting supermodular f	crossing supermodular f
out-degree	$(2,5)$	$(3,(5+\min\{4,f_{\max}\}))$ $(7/2,(5+\min\{7/2,f_{\max}\}))$ $(4,(5+\min\{3,f_{\max}\}))$
in-degree	$(3/2,\min\{7/2,f_{\max}\})$ $(2,\min\{3,f_{\max}\})$	$(3,(5+\min\{4,f_{\max}\}))$ $(7/2,(5+\min\{7/2,f_{\max}\}))$ $(4,(5+\min\{3,f_{\max}\}))$
both out- and in-degree	$(2,6,\min\{6,f_{\max}\})$ $(3,5,\min\{8,f_{\max}\})$	$(4,(6+\min\{6,f_{\max}\}),(6+\min\{6,f_{\max}\}))$ $(6,(5+\min\{8,f_{\max}\}),(5+\min\{8,f_{\max}\}))$

Table 1.1 shows the previous approximation bounds of polynomial-time algorithms and our new bounds for the DWDCN problems with weighted out-degree constraints or weighted

in-degree constraints, or both weighted out- and in-degree constraints under intersecting or crossing supermodular connectivity requirements. For the DWDCN problems with either weighted out-degree or weighted in-degree constraints, an (α,β) approximation bound of a polynomial-time algorithm means that the computed subgraph satisfies the specified connectivity requirements, has the cost at most α times the optimal cost and the degree bounds are violated by up to a factor of β. For example, we show a $(2,5)$-approximation bound for the DWDCN problem with weighted out-degree constraints under intersecting supermodular connectivity requirements. This means that the computed f-connected subgraph H of the input graph G has cost at most twice the optimal cost and violates the weighted out-degree constraints by at most a factor of 5. This improves the previous best approximation bounds of $(2,7)$ and $(3,6)$ shown by Nutov [50, 52].

For the DWDCN problems with both weighted out- and in-degree constraints, we consider $(\alpha,\beta^{out},\beta^{in})$-approximation algorithms. Such an algorithm computes an f-connected subgraph H of the input graph G, which has cost at most α times the optimal cost, violates the weighted out-degree constraints by at most a factor of β^{out}, and violates the weighted in-degree constraints by at most a factor of β^{in}. For example, we show a $(2,6,\min\{6,f_{\max}\})$-approximation algorithm for the DWDCN problem with both weighted out- and in-degree constraints under intersecting supermodular connectivity requirements f, where $f_{\max}=\max\{f(S):S\subseteq V\}$. This improves the previous best approximation bound of $(2,7,\min\{6,f_{\max}\})$. We also obtain some improvements of the previous bounds in the special case of uniform weights in the degree constraints. These results are not given in Table 1.1, but are summarised in Chapter 3.

The MNL problems have two variants depending on the type of the required output. In the *single topology* variant, the same routing topology is used for each communication round (that is, for each execution of the given communication task) whereas in the *multiple topology* variant, the routing topologies can be different in different rounds.

Our preliminary results for the MNL problems were published in [40, 41] and were based on the results of the DWDCN problems given in [52]. We further improve the approximation

guarantees for the MNL problems to the values as shown in Table 1.2 using our new approximation bounds for the DWDCN problems. The values in the table denote the number of communication rounds, which can be computed in polynomial time. The k_{opt} denotes the maximum number of rounds that can be performed. We note that our results and the previous results in the table guarantee the number of rounds $\lfloor k_{opt}/\beta \rfloor$ only for the inputs such that $w(u,v) \leq B(u)/\beta$, for each edge (u,v).

Table 1.2 Previous results and our results for the MNL problems.

Previous results				
Type of solution	unicast	convergecast	broacast	mixedcast
Single Topology	k_{opt} [61]	k_{opt} [61]	$\lfloor k_{opt}/25 \rfloor$ [54]	$\lfloor k_{opt}/36 \rfloor$ [54]
Multiple Topology	$\lfloor k_{opt}/16 \rfloor$[51] $1/31 \cdot k_{opt}$[51]	$\lfloor k_{opt}/16 \rfloor$ [54] $1/31 \cdot k_{opt}$[54]	$\lfloor k_{opt}/36 \rfloor$[54]	$\lfloor k_{opt}/100 \rfloor$[54]
Our results				
Type of solution	unicast	convergecast	broacast	mixedcast
Single Topology	-	-	$\lfloor k_{opt}/5 \rfloor$	$\lfloor k_{opt}/5 \rfloor$
Multiple Topology	$\lfloor k_{opt}/3 \rfloor$ $1/5 \cdot k_{opt}$	$\lfloor k_{opt}/3 \rfloor$ $1/5 \cdot k_{opt}$	$\lfloor k_{opt}/5 \rfloor$	$\lfloor k_{opt}/5 \rfloor$

1.4 The contents of the thesis

This thesis consists of ten chapters, tackling two classes of problems: *Directed Weighted Degree Constrained Network Design* (DWDCN) problems and the *Maximum Network Lifetime* (MNL) problems in wireless ad-hoc networks. The DWDCN problems are considered in Chapters 3 to 5, and the MNL problems are considered in Chapters 6 to 9.

In Chapter 2, we provide the basic notation and definitions used throughout this thesis. We also provide some facts from linear programming, which we need for DWDCN approximation algorithms.

In Chapter 3, we give formal definitions of the DWDCN problems which we consider in this thesis, discuss previous related results, and summarise our contribution to DWDCN approximation algorithms. Furthermore, we review Nutov's approximation algorithms for the DWDCN problems [50, 52] and give some notions and definitions, which are used in later chapters in the analysis of these algorithms. In Chapter 4, we consider the DWDCN problems under intersecting supermodular connectivity requirements and derive new improved approximation bounds. The DWDCN problems with crossing supermodular connectivity requirements are discussed in Chapter 5.

In Chapter 6, we formally define the MNL problems, discuss previous related results and summarise our contribution to MNL approximation algorithms. Furthermore, we discuss the computational complexity of the MNL problems and extend previous NP-hardness proofs of the MNL problems to some variants of these problems considered in this thesis. In Chapter 7, we consider the MNL unicast, broadcast, and convergecast problems and present approximation algorithms. Moreover, we show our improved approximation bounds for these problems. We also describe some implementation details of the MNL broadcast and convergecast approximation algorithms, which are needed to achieve polynomial running times. The MNL mixedcast problem is considered separately in Chapter 8.

In Chapter 9, we present our experimental evaluation of the MNL broadcast approximation algorithm, comparing the quality of the computed solutions with upper bounds, obtained by linear programming relaxations, and with solutions obtained by a natural heuristic. Moreover, we propose a method for improving the practical performance of the MNL approximation algorithm.

In Chapter 10, we give concluding remarks and discuss possible future research directions in the area of degree-bounded network design problems, highlighting some open problems.

Chapter 2

Preliminaries

In this chapter, we provide a formal definition of a graph which we consider and introduce notations and basic facts used in this thesis (Section 2.1). In addition, we also introduce basic terminology and facts about linear programming (LP) that are needed for the subsequent chapters.

2.1 Basic definitions for graph

Graphs considered in this thesis are always directed graphs (V,E), where V is a set of n nodes and $E \subseteq V \times V$ is a set of m directed edges. For a subgraph $H = (V,F)$ of a directed graph $G = (V,E)$, where $F \subseteq E$, and a subset S of nodes V, let $\delta_H^{out}(S)$ (resp. $\delta_H^{in}(S)$) denote the set of edges in H leaving (resp. entering) S. That is, for example, $\delta_H^{out}(S) = \{(u,v) \in H : u \in S, v \in V \backslash S\}$. For simplicity of notation, for a node v we will write $\delta_H^{out}(v)$ instead of $\delta_H^{out}(\{v\})$. For a function $x : E \to \mathbb{R}^+$, where $\mathbb{R}^+$ denotes the set of non-negative real numbers, and a subgraph $H = (V,F)$, we define $x(H) = x(F) = \sum_{e \in F} x(e)$. We will normally have a *cost function* $c : E \to \mathbb{R}^+$ defined on the edges of G and *degree bounds* $b : B \to \mathbb{R}^+$ defined on the nodes of a given subset $B \subseteq V$. In the case of *weighted degree bounds*, we also have edges

weights $w : E \to \mathbb{R}^+$. The cost of a subgraph H of G is defined as $c(H)$ and we say that a subgraph H satisfies the weighted out-degree bounds if $w(\delta_H^{out}(v)) \leq b(v)$ for all nodes $v \in B$.

For a set-function f on V, that is, $f : 2^V \to \mathbb{R}^+$, we say that a subgraph H of G is *f-connected*, if $|\delta_H^{in}(S)| \geq f(S)$ for every $\emptyset \neq S \subsetneq V$. For two sets $X, Y \subseteq V$, we say that X and Y *intersect* if $X \cap Y \neq \emptyset$, and *cross* if all sets $X \cap Y$, $X \setminus Y$, $Y \setminus X$, $V \setminus (X \cup Y)$ are non-empty. A set function f is called *intersecting supermodular (resp. crossing supermodular)* if each pair of sets $X, Y \subseteq V$ that intersect (resp. cross) satisfies the condition of supermodularity:

$$f(X) + f(Y) \leq f(X \cap Y) + f(X \cup Y).$$

A family $\mathscr{F}$ of sets is called *laminar*, if for every two sets $X, Y \in \mathscr{F}$, either $X \cap Y = \emptyset$, or $X \subseteq Y$, or $Y \subseteq X$.

We say that a set of paths from a node r to a node v is edge-disjoint, if no two paths have a common edge. A directed graph G is said to be *k-edge-outconnected with root r*, if there are k edge-disjoint paths from node r to every node $v \neq r$ in G. Similarly, a directed graph G is said to be *k-edge-inconnected with root r*, if for each node $v \neq r$, there are k edge-disjoint paths from v to r in G. For example, the graph in Figure 2.1 is 2-*edge-outconnected with root r* because for each node v (except root r) there are 2 edge-disjoint paths from r to v. A graph is strongly k-connected if for every pair of nodes v, u, there are k node-disjoint paths from v to u and k node-disjoint paths from u to v.

An *out-arborescence* (a broadcast tree) T_{out} is a directed spanning tree that has a unique path from a root r to every node. An *in-arborescence* (a convergecast tree) T_{in} is a directed spanning tree that has a path from every node to the root r. An *arborescence* refers to either out-arborescence or in-arborescence, depending on the context.

There are k edge-disjoint paths from r to v, if and only if, there is an r-v flow of value k with unit edge capacities. By the maximum flow minimum cut theorem, this also means that there are k edge-disjoint paths from r to v, if and only if, the capacity of each cut is at least k. Hence,

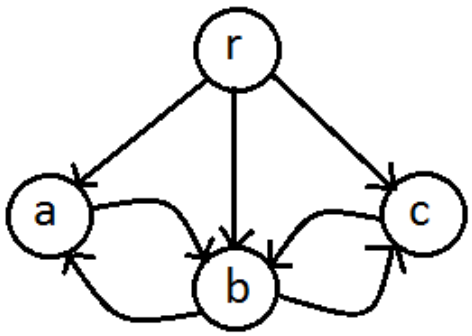


Figure 2.1 Example of a graph, which is 2-edge-outconnected with root r: two edge-disjoint paths from r to each of the nodes a, b, c.

there are k edge-disjoint path from r to v, if and only if, $\delta^{in}_G(S) \geq k$ for each subset of nodes S such that $v \in S \subseteq V \backslash \{r\}$. Thus, there are k edge-disjoint paths from a node r to all other nodes $v \in V \backslash \{r\}$, if and only if, $\delta^{in}_G(S) \geq k$, for every subset of nodes S such that $\emptyset \neq S \subseteq V \backslash \{r\}$.

We should mention that there are important network design problems with connectivity defined by a function which is not supermodular. One example is the *directed Steiner Tree* problem of finding a minimum cost directed out-tree rooted at r that contains a given subset of nodes $D \subseteq V$. Such connectivity requirements can be specified by the set function f, where $f(S) = 1$, if $r \notin S$ and $S \cap D \neq \emptyset$ and $f(S) = 0$ otherwise. This function f however is not crossing or intersecting supermodular, but has the weakly supermodular property. A set function f is *weakly supermodular* if for every two subsets X and Y,

$$f(X) + f(Y) \leq f(X \cap Y) + f(X \cup Y), \text{ or}$$
$$f(X) + f(Y) \leq f(X - Y) + f(Y - X).$$

2.2 Linear programs and integer programs

A linear programming problem is to optimise a linear objective function, subject to linear equality and/or linear inequality constraints. A minimisation linear program (LP) can be

expressed in the standard (matrix) form as

Minimise:	$c^T x$	}	objective function
Subject to:	$Ax \leq b$	}	linear constraints
	$x \geq 0$	}	non-negative constraints

where vector $c = (c_1, \ldots, c_n) \in \mathbb{R}^n$ and matrix $A = (a_{i,j} : i = 1, \ldots, m, j = 1, \ldots, n) \in \mathbb{R}^{m \times n}$ and $b = (b_1, \ldots, b_m)$ are given and $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is a vector of n real variables.

If $x \in \mathbb{R}^n$ satisfies all linear constraints specified in an LP, then x is called a *feasible solution* to this LP. We say that an LP is *feasible* if it has at least one feasible solution x, and *infeasible* otherwise. The *feasible region* of an LP is the set of all possible $x \in \mathbb{R}^n$ that satisfy the specified linear constraints, that is, the set of all possible feasible solutions. An *optimal solution* x^* to an LP is a feasible solution, which has the minimum objective function value in the case of a minimisation problem or that has the largest objective value in the case of a maximisation problem. That is, for a minimisation problem, $c^T x^* = \min\{c^T x : Ax \leq b \text{ and } x \geq 0\}$. An LP may have more than one optimal solution, but there is only one optimal solution value. Given a constraint, $ax \leq b$ in an LP, we say that the constraint is active (tight) for $x' \in \mathbb{R}^n$, if $ax' = b$. A *basic feasible solution* to an LP is a feasible solution x that has n linearly independent active (tight) constraints. An *optimal basic feasible solution* is a basic feasible solution that has an optimal objective value.

The set of all feasible solutions (feasible region) to an LP forms a *polyhedron* $P = \{x \in \mathbb{R}^n : Ax \leq b \text{ and } x \geq 0\}$. A bounded polyhedron is called a *polytope*. Thus we refer to the feasible region of an LP as an *LP-polyhedron* or an *LP-polytope*, and we refer to feasible solutions as feasible points. An *extreme point* is a vertex point of a polyhedron: a point which is not a strict linear combination of any two points in the polyhedron. The basic feasible solutions of an LP correspond to the extreme points of the LP-polyhedron. If an LP is feasible, then it has an optimal solution at an extreme point of the feasible region. In other words, if there is an optimal solution to an LP, then there is an optimal basic feasible solution.

The main methods for solving linear programs includes the simplex method [9] (and its many variants), the interior point method [25, 31] and the ellipsoid method [32]. The ellipsoid method and the interior point method solve linear programs in polynomial time. In general, the interior-point method and the simplex method are more efficient than the ellipsoid method in practice. However, the ellipsoid method has an advantage that it can be applied to a more general setting.

When we apply the ellipsoid method, we do not need to know the constraints explicitly. We only need to have access to a *separation oracle*. A separation oracle is a polynomial time algorithm that determines whether a given candidate point x is feasible or not, and finds a violated constraint if x is not feasible. Hence, the ellipsoid method can be employed for solving linear programs in polynomial-time (in terms of the number of variables), even if there are exponentially many constraints, provided that we have a separation oracle. In this thesis, the linear programs which appear in algorithms for networks design problems and MNL problems have exponentially many constraints. Therefore, polynomial time separation oracles are needed in order to claim that such LPs can be solved in polynomial time. We will give details of these LPs and corresponding separation oracles later in this thesis.

Chapter 3

Directed Weighted Degree Constrained Network Design (DWDCN)

In Chapters 3 – 5, we discuss Directed Weighted Degree Constrained Network Design (DWDCN) problems and show how we derive new improved approximation bounds for some class of DWDCN problems. As mentioned in Section 1.1, the DWDCN problems have many variants depending on the type of the connectivity requirements and on the type of the degree bounds. In this thesis, we consider DWDCN problems with the following types of the degree bounds:

- DWDCN with weighted out-degree constraints,
- DWDCN with weighted in-degree constraints,
- DWDCN with weighted out- and in-degree constraints.

We consider these types of DWDCN problems under intersecting or crossing supermodular connectivity requirements, i.e, a connectivity requirement is specified by an intersecting or crossing supermodular function.

We follow the approach proposed by Nutov [52], who developed polynomial bi-criteria approximation algorithms for these problems. By conducting a detailed analysis of Nutov's approximation algorithms, we improve a number of previous approximation bounds. In Chapter 4, we consider DWDCN problems under intersecting supermodular connectivity requirements and derive new improved approximation bounds for these problems. The problems with crossing supermodular connectivity requirements are discussed separately in Chapter 5.

This chapter consists of the following parts. In Section 3.1, we formally define the DWDCN problems which we consider in this thesis. In Sections 3.2 and 3.3, we summarise our contribution to the DWDCN problems and discuss previous related work. In Section 3.4, we review Nutov's approximation algorithms for the DWDCN problems [52]. In Section 3.5, we give some notions and definitions, which will be used in Chapters 4 and 5 in the analysis of these approximation algorithms.

3.1 Definitions of problems and approximation bounds

Let $G = (V, E, c, w)$ denote a directed graph (V, E) with an edge-cost function $c : E \to \mathbb{R}^+$ and an edge-weight function $w : E \to \mathbb{R}^+$. The problem with out-degree constraints is formally defined as follows.

Problem: DWDCN problem with weighted out-degree constraints
Instance: $G = (V, E, c, w)$, a subset $B \subseteq V$, out-degree bounds $b : B \to \mathbb{R}^+$, and an intersecting (or, crossing) supermodular set function f on V.
Objective: Find a minimum cost f-connected subgraph $H = (V, F)$ of G which satisfies the weighted out-degree constraints

$$w(\delta_H^{out}(v)) \leq b(v), \text{ for all } v \in B. \tag{3.1}$$

The DWDCN problem with weighted in-degree constraints is defined analogously, with the weighted out-degree constraints (3.1) replaced with the weighted in-degree constraints

$$w(\delta_H^{in}(v)) \leq b^{in}(v), \text{ for all } v \in B^{in}, \tag{3.2}$$

where $b^{in} : B^{in} \to \mathbb{R}^+$ are in-degree bounds for the nodes in a given subset $B^{in} \subseteq V$.

In the case when we have both out- and in-degree constraints, the input includes both degree bound functions $b : B \to \mathbb{R}^+$ and $b^{in} : B^{in} \to \mathbb{R}^+$. The objective is to find a minimum cost f-connected subgraph $H = (V, F)$ of G which satisfies both weighted out- and in-degree constraints (3.1) and (3.2).

Various connectivity requirements can be defined by intersecting or crossing supermodular functions. For example, consider a problem of finding k-edge-outconnected subgraph H of G with root r, which satisfies the weighted out-degree constraints (3.1). This problem is known as the *Weighted degree constrained k-edge-outconnected subgraph* problem and it is a special case of the DWDCN problems with weighted out-degree constraints. The connectivity requirement of this problem can be defined by the intersecting supermodular function f such that $f(S) = k$ for all $\emptyset \neq S \subseteq V \setminus \{r\}$ and $f(S) = 0$ otherwise. An example of a problem associated with a crossing supermodular function is the *Weighted degree constrained strongly k-connected subgraph* problem. The connectivity requirement of this problem can be defined by the crossing supermodular function f such that $f(S) = k$ for all $\emptyset \neq S \subsetneq V$ and $f(\emptyset) = f(V) = 0$.

For the DWDCN problems with either (weighted) out-degree constraints or (weighted) in-degree constraints, we consider (α, g)-approximation algorithms. Such an algorithm computes an f-connected subgraph H of the input graph G, which has the cost at most α times the optimal cost and for each node $v \in B$, the degree of v in H is at most $g(b(v))$, if the input has a feasible solution (i.e, has an f-connected subgraph which satisfies the weighted degree constraints). If there is no feasible solution, then the algorithm either realises that this is the case or returns an f-connected subgraph that satisfies the degree bounds $g(b(v))$. Thus the first parameter α of (α, g)-approximation indicates the approximation ratio for the cost whereas

the second parameter g indicates the approximation of the degree bounds. The function g would typically be of a form $\beta x+\gamma$, for some $\beta \geq 1$ and $\gamma \geq 0$. In this case, instead of writing "$(\alpha, \beta x+\gamma)$-approximation", we will write "$(\alpha, \beta b(v)+\gamma)$-approximation" (to follow the notation from the previous papers).

For the DWDCN problems with both (weighted) out- and in-degree constraints, we extend the definition of an (α, g)-approximation algorithm to cover both in- and out-degree bounds. An algorithm for the DWDCN problem with both (weighted) out- and in-degree constraints is said to be (α, g', g'')-approximation if, for a feasible input, the algorithm returns f-connected subgraph H of G with cost at most α times the optimal cost, for each node $v \in B$, the (weighted) out-degree of v is at most $g'(b(v))$ and for each node $v \in B^{in}$, the (weighted) in-degree of v is at most $g''(b^{in}(v))$.

3.2 Our contribution

We consider the approximation algorithms for the DWDCN problems under intersecting supermodular connectivity requirements, proposed by Nutov [50, 52]. By developing more detailed analysis of these algorithms, we derive better approximation bounds. Our bounds are stated in Theorems 3.1–3.5 below and compared with Nutov's bounds in Tables 3.1 and 3.2. The results for crossing supermodular requirements are obtained by a reduction to the intersecting supermodular requirements.

Under intersecting supermodular connectivity requirements, Nutov [50, 52] gave polynomial-time $(2, 7b(v))$- and $(3, 6b(v))$-approximation algorithms for the DWDCN problem with weighted out-degree constraints. For the case of unit weights, he gave a polynomial-time $(2, 2b(v)+4)$-approximation algorithm. Later in [53] he improved and generalised this bound to $(1/\varepsilon, \lceil b(v)/(1-\varepsilon)\rceil+3)$-approximation, where $\varepsilon \in [0, 1/2)$. Our bounds for these problems are given in the following theorem.

Theorem 3.1. *Under intersecting supermodular connectivity requirements, the* DWDCN *problem with weighted out-degree constraints admits a polynomial-time* $(2, 5b(v))$*-approximation algorithm. For the case of unit weights, the problem admits a polynomial-time* $(2,\ 2b(v)+2)$*-approximation algorithm.*

For DWDCN with intersecting supermodular connectivity requirements and in-degree constraints, Nutov [50, 52] gave a polynomial-time $(1,\ \min\{4, f_{\max}\} \cdot b^{in}(v))$-approximation algorithm, where $f_{\max} = \max_{S \subseteq V} f(S)$. For unit weights, Nutov [52] also showed that the problem admits an exact polynomial-time $(1,\ \min\{f_{\max}, b^{in}(v)\})$-algorithm. Our bounds are given in the following theorem.

Theorem 3.2. *Under intersecting supermodular connectivity requirements, the* DWDCN *problem with weighted in-degree constraints admits polynomial-time* $(2,\ \min\{3, f_{\max}\} \cdot b^{in}(v))$ *and* $(3/2,\ \min\{7/2, f_{\max}\} \cdot b^{in}(v))$*-approximation algorithms.*

Our results for the DWDCN problem with weighted in-degree constraints under intersecting supermodular connectivity requirements stated in Theorem 3.2 improve the approximation of the weighted degree bounds, paying for this with an increased approximation ratio for the cost. The algorithm for this problem can be applied to solve *Maximum Network Lifetime Convergecast problem* [40, 41, 54], where the weighted degree bounds are specified, but there are no edge cost. Our Theorem 3.2 gives a better approximation bound for this problem than in Nutov [54, 55].

Under crossing supermodular connectivity requirements, Nutov's method [52] gives the same approximation bounds for both out-degree constraints and in-degree constraints (that is, for both cases when either all degree constraints refer to out-degrees or all degree constraints refer to in-degrees). For the weighted case, Nutov [52] gave polynomial-time $(3,\ (7+\min\{4, f_{\max}\}) \cdot b(v))$ and $(4,\ (6+\min\{4, f_{\max}\}) \cdot b(v))$-approximation algorithms. For unit weights, he also gave a polynomial-time $(3,\ 2b(v)+4+\min\{f_{\max}, b(v)\})$-approximation algorithm. The following theorem summaries our results for these problems.

Theorem 3.3. *Under crossing supermodular connectivity requirements, the* DWDCN *problem with weighted out-degree constraints admits polynomial-time* $(3,\ (5+\min\{4, f_{\max}\})\cdot b(v))$, $(4,\ (5+\min\{3, f_{\max}\})\cdot b(v))$, *and* $(7/2,\ (5+\min\{7/2, f_{\max}\})\cdot b(v))$*-approximation algorithms. For the case of unit weights, the problem admits a polynomial-time* $(3,\ 2b(v)+2+\min\{f_{\max}, b(v)\})$*-approximation algorithm. The same bounds apply for the cases with in-degree constraints.*

All four approximation bounds given in Theorem 3.3 are better than the corresponding bounds in Nutov [52].

For the DWDCN problem with both weighted out- and in-degree constraints (that is, one input instance can include both out- and in-degree constraints) and the intersecting supermodular connectivity requirements, Nutov [52] gave polynomial-time $\left(2,\ 7b(v),\ \min\{6, f_{\max}\}\cdot b^{in}(v)\right)$ and $\left(3,\ 6b(v),\ \min\{7, f_{\max}\}\cdot b^{in}(v)\right)$-approximation algorithms. For unit weights, he also showed that the problem admits a polynomial-time $(2,\ 2b(v)+4,\ \min\{2b^{in}(v)+2, f_{\max}\}$-approximation algorithm. Our bounds for these problem are given in the following theorem.

Theorem 3.4. *Under intersecting supermodular connectivity requirements, the* DWDCN *problem with weighted out- and in-degree constraints admits polynomial-time* $(2,\ 6b(v),\ \min\{6, f_{\max}\}\cdot b^{in}(v))$ *and* $(3,\ 5b(v),\ \min\{8, f_{\max}\}\cdot b^{in}(v))$*-approximation algorithms. For the case of unit weights, the problem admits a polynomial-time* $(2,\ 2b(v)+3,\ \min\{2b^{in}(v)+2, f_{\max}\}$*-approximation algorithm.*

Under the crossing supermodular connectivity requirements, Nutov [52] gave polynomial-time $(4,\ (7+\min\{6, f_{\max}\})\cdot b(v),\ (7+\min\{6, f_{\max}\})\cdot b^{in}(v))$ and $(6,\ (6+\min\{7, f_{\max}\})\cdot b(v),\ (6+\min\{7, f_{\max}\})\cdot b^{in}(v))$-approximation algorithms. For the case of unit weight, he also showed that the problem admits a polynomial-time $(4,\ 2b(v)+4+\min\{2b(v)+2, f_{\max}\},\ 2b^{in}(v)+4+\min\{2b^{in}(v)+2, f_{\max}\})$-approximation algorithm. The following theorem gives our bounds for these problems.

Theorem 3.5. *Under crossing supermodular connectivity requirements, the* DWDCN *problem with weighted out- and in-degree constraints admits polynomial-time* $(4,\ (6+\min\{6, f_{\max}\})\cdot b(v),\ (6+\min\{6, f_{\max}\})\cdot b^{in}(v))$ *and* $(6,\ (5+\min\{8, f_{\max}\})\cdot b(v),\ (5+\min\{8, f_{\max}\})\cdot b^{in}(v))$*-approximation algorithms. For the case of unit weights, the problem admits a polynomial-time* $(4,\ 2b(v)+3+\min\{2b(v)+2, f_{\max}\},\ 2b^{in}(v)+3+\min\{2b^{in}(v)+2, f_{\max}\})$*-approximation algorithm.*

We note that Nutov [52] states the $\left(3,\ 6b(v),\ \min\{7, f_{\max}\}\cdot b^{in}(v)\right)$-approximation bounds for the intersecting supermodular connectivity requirements, but as far as we can see, the proof given there supports only a weaker bound of $\left(3,\ 6b(v),\ \min\{8, f_{\max}\}\cdot b^{in}(v)\right)$, and consequently a weaker bound of $(6,\ (6+\min\{8, f_{\max}\})\cdot b(v),\ (6+\min\{8, f_{\max}\})\cdot b^{in}(v))$ for the crossing supermodular connectivity requirements.

Table 3.1 (α, g)-approximations for DWDCN problem with (weighted) out-degree or in-degree constraints (but not both) under intersecting and crossing supermodular connectivity requirements. The parameter $\varepsilon \in [0, 1/2)$.

Previous results			
	intersecting supermodular		crossing supermodular
w	out-degree	in-degree	out-degree or in-degree
any	$(2,\ 7b(v))$ [52] $(3,\ 6b(v))$ [52]	$\left(1,\ \min\{4, f_{\max}\}b^{in}(v)\right)$ [52]	$(3,\ (7+\min\{4, f_{\max}\})\,b(v))$ [52] $(4,\ (6+\min\{4, f_{\max}\})\,b(v))$ [52]
unit	$(2,\ 2b(v)+4)$ [52] $(\frac{1}{\varepsilon}, \lceil\frac{b(v)}{1-\varepsilon}\rceil+3)$ [53]	$\left(1,\ \min\{f_{\max}, b^{in}(v)\}\right)$ [52]	$\left(3,\ 2b(v)+3+\min\{f_{\max}, b^{in}(v)\}\right)$ [52]
Our contribution			
	intersecting supermodular		crossing supermodular
w	out-degree	in-degree	out-degree or in-degree
any	$(2,\ 5b(v))$	$\left(\frac{3}{2},\ \min\{\frac{7}{2}, f_{\max}\}b^{in}(v)\right)$ $\left(2,\ \min\{3, f_{\max}\}b^{in}(v)\right)$	$(3,\ (5+\min\{4, f_{\max}\})\,b(v))$ $\left(7/2, \left(5+\min\{\frac{3}{2},\ f_{\max}\}\right)b(v)\right)$ $(4, (5+\min\{3,\ f_{\max}\})\,b(v))$
unit	$(2, 2b(v)+2)$		$\left(3,\ 2b(v)+2+\min\{f_{\max}, b^{in}(v)\}\right)$

3.3 Previous related work

Jain [27] introduced the *iterative rounding* method for network design problems on undirected graphs and gave a 2-approximation algorithm for the *minimum cost Steiner network problem*. Lau *et al.* [38] extended the method to solve the *degree bounded survivable network design* problem for undirected graphs. This extended method was called the *iterative relaxation* and has been applied to various network design problems with degree bounded constraints, including spanning tree [1, 62], Steiner network [62, 46, 36], edge-connectivity [50, 52, 1] and node-connectivity [53, 14, 34]. Mostly, these problems are considered in undirected graphs.

Frank [16] showed that the network design problem (without degree constraints) of finding minimum cost f-connected subgraph can be solved optimally in polynomial-time if function f is intersecting supermodular. However, there are network design problems with a crossing supermodular function f (without degree constraints) known to be NP-hard. Melkonian and Tardos [49] gave a 2-approximation algorithm for problems with crossing supermodular functions based on Frank's result of [16].

Lau *et al.* [38] were the first to consider degree-bounded network design problems with (edge) general connectivity requirements and unit weights. For undirected graphs, they gave a polynomial-time $(2,\ 2b(v)+3)$-approximation algorithm under the weakly supermodular connectivity requirements. For directed graph, they also provided a polynomial-time $(4, 4b(v)+6, 4b^{in}(v)+6)$-approximation algorithm under intersecting supermodular connectivity requirements, and a $(8, 8b(v)+12, 8b^{in}(v)+12)$-approximation algorithm under crossing supermodular connectivity requirements. For crossing supermodular connectivity requirements, they showed in [39] an improved bound of $(3, 3b(v)+5, 3b^{in}(v)+5)$. For $0,1$-valued intersecting supermodular connectivity requirements, they also give a $(2, 2b(v)+2, b^{in}(v))$-approximation algorithm. Later, Lau *et al.* [35] gave $(2, b(v)+3)$ approximation algorithm for degree-bounded Steiner forest and $(2, 6r_{\max}+3)$ for degree-bounded survival network design problem, where $r_{\max}$ is the maximum connectivity requirement over all pairs of nodes.

Table 3.2 (α, g', g'')-approximations for DWDCN problem with (weighted) out-degree and in-degree constraints under intersecting and crossing supermodular connectivity requirements. The parameter $\varepsilon \in [0, 1/2]$ and $f_{\max} = \max_{S \subseteq V} f(S)$ is the maximum f-value.

Previous results		
w	intersecting supermodular	crossing supermodular
any	$(2, 7b(v), \min\{6, f_{\max}\}b^{in}(v))$ [52] $(3, 6b(v), \min\{8, f_{\max}\}b^{in}(v))$ [52]	$(4, (7+\min\{6, f_{\max}\})b(v)), (7+\min\{6, f_{\max}\})b^{in}(v))$ [52] $(6, (6+\min\{8, f_{\max}\})b(v)), (6+\min\{8, f_{\max}\})b^{in}(v))$ [52]
unit	$(2, 2b(v)+4, \min\{2b^{in}(v)+2, f_{\max}\})$[52] $(1/\varepsilon, \lceil \frac{b(v)}{1-\varepsilon} \rceil + 4, \lceil \frac{b^{in}(v)}{1-\varepsilon} \rceil + 4)$ [1]	$(4, 2b(v)+4+\min\{2b(v)+2, f_{\max}\}), 2b(v)+4+\min\{2b(v)+2, f_{\max}\})$ [52] $(2/\varepsilon, \lceil \frac{b(v)}{(1-\varepsilon)} \rceil + 4 + f_{\max}, \lceil \frac{b^{in}(v)}{(1-\varepsilon)} \rceil + 4 + f_{\max})$ [1] $(3, 3b(v)+5, 3b^{in}(v)+5)$ [39]
Our contribution		
w	intersecting supermodular	crossing supermodular
any	$(2, 6b(v), \min\{6, f_{\max}\}b^{in}(v))$ $(3, 5b(v), \min\{8, f_{\max}\}b^{in}(v))$	$(4, (6+\min\{6, f_{\max}\})b(v), (6+\min\{6, f_{\max}\})b^{in}(v))$ $(6, (5+\min\{8, f_{\max}\})b(v), (5+\min\{8, f_{\max}\})b^{in}(v))$
unit	$(2, 2b(v)+3, \min\{2b^{in}(v)+2, f_{\max}\})$	$(4, 2b(v)+3+\min\{2b(v)+2, f_{\max}\}, 2b^{in}(v)+3+\min\{2b^{in}(v)+2, f_{\max}\})$

Bansal *et al.* [1] showed a polynomial-time $(1/\varepsilon, \lceil b(v)/(1-\varepsilon)\rceil + 4, \lceil b^{in}(v)/(1-\varepsilon)\rceil + 4)$-approximation algorithm for directed graphs, unit weights, and intersecting supermodular connectivity requirements. For crossing supermodular connectivity requirements, they gave a polynomial-time $(2/\varepsilon, \lceil b(v)/(1-\varepsilon)\rceil + 4 + f_{\max}, \lceil b^{in}(v)/(1-\varepsilon)\rceil + 4 + f_{\max})$-approximation algorithm. In both cases, $0 \leq \varepsilon \leq 1/2$.

For the case of the weighted degree constraints, Fukunaga and Nagamochi [17] considered the problems in undirected graphs and showed a $(2,\ 7b(v))$-approximation algorithm for weakly supermodular connectivity requirements. Nutov [50, 52] considered network design problems with weighted degree constraints in directed graphs and his results are given in Section 3.2. All results mentioned in this section, as well as our new results, are based on the general idea of iterative rounding, which originated in Jain [27] and was later adapted to degree-bounded network design in Lau *et al.* [38] and Singh *et al.* [62].

3.4 The Iterative Relaxation approximation algorithmic framework for DWDCN problems

In this section, we review the Nutov's approximation algorithms [52] for the DWDCN problems. These algorithms give the previous best approximation bounds for DWDCN with weighted degree constraints considered in this thesis. We obtain our improved approximation bounds by developing more detailed analysis of this algorithmic framework. Our description refers to the DWDCN problem with weighted out-degree constraints. There are analogous algorithms for the cases of in-degree constraints and both out- and in-degree constraints.

The algorithm is based on the iterative relaxation, which works as follows. First the DWDCN problem is formulated as an integer program and its linear programming (LP) relaxation is solved. The LP-relaxation is to minimise $\sum_{e\in E} c(e)\cdot x(e)$ over the polytope $P(f,b)$

defined by the following constraints:

$$x\left(\delta_E^{in}(S)\right) \geq f(S), \quad \text{for all} \quad \emptyset \neq S \subsetneq V, \tag{3.3}$$

$$\sum_{e \in \delta_E^{out}(v)} x(e)w(e) \leq b(v), \quad \text{for all} \quad v \in B, \tag{3.4}$$

$$0 \leq x(e) \leq 1, \quad \text{for all} \quad e \in E. \tag{3.5}$$

The algorithm checks whether the LP-polytope $P(f,b)$ is empty or not. If the polytope is empty (not feasible), then algorithm returns "Infeasible" and terminates. Clearly, the corresponding integer program has no feasible solution as well. Otherwise, the algorithm performs the following iterative process.

First compute an optimal basic feasible solution $x = (x(e))_{e \in E}$. Based on the values $x(e)$, certain edges are removed from E and some of them are added to the (partial) solution $J \subseteq E$; initially $J = \emptyset$. More specifically, edges e with $x(e) = 0$ are removed from E, while edges with $x(e) \geq 1/\alpha$ are removed from E and added to the partial solution J, where α is a fixed parameter. Thus, only edges e with $0 < x(e) < 1/\alpha$ are left in set E. The algorithm maintains the set B' of the nodes with weighted out-degree bounds; initially, $B' = B$. A node v is removed from B', if the degree of v is less than or equal to $\triangle$, where $\triangle$ is another fixed parameter. That is, we remove from B' all nodes with $|\delta_E^{out}(v)| \leq \triangle$. Note that the nodes removed from B' remain in the graph.

With given α and the updated sets E, B' and J, the residual LP problem of minimising $\sum_{e \in E} c(e) \cdot x(e)$ over the polytope $P(f,b;\alpha,J,B')$ is then solved in the next iteration. The residual polytope $P(f,b;\alpha,J,B')$ is defined by the following constraints:

$$x\left(\delta_E^{in}(S)\right) \geq f_J(S) \quad \equiv f(S) - |\delta_J^{in}(S)|, \quad \text{for all} \quad \emptyset \neq S \subsetneq V, \tag{3.6}$$

$$\sum_{e \in \delta_E^{out}(v)} x(e)w(e) \leq b_{\alpha,J}(v) \equiv b(v) - w(\delta_J^{out}(v))/\alpha, \quad \text{for all} \quad v \in B', \tag{3.7}$$

$$0 \leq x(e) \leq 1, \quad \text{for all} \quad e \in E. \tag{3.8}$$

Initialization: $J \leftarrow \emptyset$, $B' \leftarrow B$, $E \leftarrow E \setminus \{(u,v) \in E : w(u,v) > b(v)\}$;
if $P(f,b) = \emptyset$ **then** return "INFEASIBLE" and STOP;
while $E \neq \emptyset$ **do**
    Find an optimal basic feasible solution
    $x = \min\{\sum_{e \in E} c(e) \cdot x(e) : x \in P(f,b;\alpha,J,B')\}$.
    Remove from E all edges with $x(e) = 0$.
    Remove from E all these edges all edges with $x(e) \geq 1/\alpha$ and add them to J.
    Remove from B' all nodes $v \in B$ with $|\delta_E^{out}(v)| \leq \triangle$.
end
return $F \leftarrow J$

Figure 3.1 Algorithm for DWDCN problem with weighted out-degree constraints under intersecting supermodular connectivity requirements

Observe that $P(f,b;\alpha,\emptyset,B)$ is the polytope $P(f,b)$ used in the first iteration. More generally, the residual LP problem are of the same type as the initial LP problem. If $P_G(f,b)$ denotes the initial LP-polytope for the input graph G and $P_{G'}(f,b;\alpha,J,B')$ denotes the residual LP polytope for the current graph G', then $P_{G'}(f,b;\alpha,J,B') = P_{G'}(f',b')$, where J is the partial solution (selected edges) obtained from G , $f'(S) = f(S) - |\delta_J^{in}(S)|$, $b' = b(v) - w(\delta_J^{out}(v))/\alpha$, and $B' \subseteq B$ is the set of nodes with $|\delta_{G'}^{out}(v)| \geq \triangle$.

This iterative process continues until E becomes empty. The algorithm is summarised in Figure 3.1. If the initial polytope $P(f,b)$ is feasible, then the residual polytopes $P(f,b;\alpha,J,B')$ in all subsequent iterations are also feasible. The values of the parameters α and $\triangle$ are selected in such a way that it can be proven that at least one edge is removed from E in each iteration. Hence, the algorithm is guaranteed to terminate in $O(m)$ iterations. At the end of the computation, the algorithm outputs the final set J. Denoting this set by F, the subgraph $H = (V,F)$ is an f-connected subgraph of G because it can be shown that the algorithm maintains the invariant that for each $\emptyset \neq S \subsetneq V$,

$$|\delta_E^{in}(S)| \geq f(S) - |\delta_J^{in}(S)|.$$

At the end of the computation, $E = \emptyset$, so this invariant implies that for each $\emptyset \neq S \subsetneq V$, $|\delta_J^{in}(S)| \geq f(S)$, as required by the f-connectedness.

Observe that the initial LP problem $\min\{c \cdot x : x \in P(f,b)\}$ and the residual LP problems $\min\{c \cdot x : x \in P(f,b;\alpha,J,B')\}$ have at most m variables but have an exponential number of constraints, so to solve the LP problem in each iteration in polynomial-time, we need a polynomial-time separation oracle. Jain [27] showed that a separation oracle for the residual LP problems can be easily obtained from the separation oracle for the initial LP problem. Hence, we will only show the separation oracle for the initial LP problem.

The separation oracle for the initial LP problem (minimising the cost over polytope $P(f,b)$) works as follows. Let x be a given candidate point. For the weighted degree constraints (3.4), we can check one by one whether the given constraint is violated. Since $|B| \leq |V|$, we have at most n such constraints, so this can be done in $O(n^2)$ time. Now consider the cut constraints (3.3). We need to confirm that x satisfies all these constraints or find a subset S such that the constraint corresponding to S is not satisfied. For a such subset S, we would have $x(\delta_E^{in}(S)) < f(S)$. Let $h(S) = x(\delta_E^{in}(S)) - f(S)$. It is clear that a violated set exists, if and only if, $\min\{h(S) : \emptyset \neq S \subsetneq V\}$ is negative. Note that the function h is the sum of the submodular function $x(\delta_E^{in}(.))$ and intersecting submodular function $-f(.)$, so function h is intersecting submodular. This means that any pair of sets $X, Y \subseteq V$ that intersect ($X \cap Y \neq \emptyset$) satisfies the following submodularity condition:

$$h(X) + h(Y) \geq h(X \cap Y) + h(X \cup Y).$$

Schrijver [60] developed a polynomial-time algorithm to minimise submodular functions. This algorithm requires submodularity condition for each pair $X, Y \subseteq V$. Since function h is only intersecting submodular, we cannot directly apply Schrijver's algorithm to minimise h. Instead, we reduce finding $\min\{h(S) : \emptyset \neq S \subsetneq V\}$ to $O(n^2)$ submodular function minimisation. We fix $v, u \in V$, $v \neq u$, define $V_{v,u} = V \setminus \{v,u\}$, and set $h_{v,u}(S) = h(S \cup \{v\})$ for each $S \subseteq V_{v,u}$. Function $h_{v,u}$ is a set function on $V_{v,u}$. For any two sets $X \subseteq V_{v,u}$ and $Y \subseteq V_{v,u}$, sets $X \cup \{v\}$ and $Y \cup \{v\}$ intersect, so $h_{v,u}(X) + h_{v,u}(Y) \geq h_{v,u}(X \cap Y) + h_{v,u}(X \cup Y)$. Therefore, the function $h_{v,u}$ is "fully" submodular on $V_{v,u}$. Hence, we can apply Schrijver's algorithm to find $\min\{h_{v,u}(S) : S \subseteq V_{v,u}\}$.

Since

$$\begin{aligned} \min_{v,u \in V}\{\min\{h_{v,u}(S) : S \subseteq V_{v,u}\}\} &= \min_{v,u \in V}\{\min\{h(S) : v \in S \subseteq V \backslash \{v\}\}\} \\ &= \min\{h(S) : \emptyset \neq S \subsetneq V\}, \end{aligned}$$

we can find $\min\{h(S) : \emptyset \neq S \subsetneq V\}$ by applying Schrijver's algorithm $n(n-1)$ times, once for each pair v, u of distinct nodes in V. Thus polynomial-time separation oracle implies that the initial LP problem can be solved in polynomial-time using the ellipsoid method [32].

3.5 The notion of $(\alpha, \triangle)$-sparseness of DWDCN polytopes

To explain the approximation property of the algorithm, as given in [52], we use the definition of sparseness of a polytopes $P(f, b; \alpha, J, B')$ given below. This is the crucial definition in the analysis of the approximation algorithm in [52] and also in our analysis in this thesis. Similar notions have been used in [38, 62, 1, 35, 46].

Definition 1. *For $\alpha \geq 1$ and $\triangle \geq 1$, a polytope $P(f, b; \alpha, J, B')$ is $(\alpha, \triangle)$-sparse, if and only if there exists a node $v \in B'$ with $|\delta_E^{out}(v)| \leq \triangle$, or for any basic solution $x \in P(f, b; \alpha, J, B')$, there exists $e \in E$ such that $x(e) = 0$ or $x(e) \geq 1/\alpha$.*

It is well known that if an LP polytope P is feasible, then the problem $\min\{c \cdot x : x \in P\}$ has an optimal basic solution x. Nutov [52] proved the following lemma.

Lemma 3.6. *[52] If for any $J \subseteq E, B' \subseteq B$ the polytope $P(f, b; \alpha, J, B')$ is $(\alpha, \triangle)$-sparse (if non-empty), then the* DWDCN *problem for weighted out-degree constraints and in-degree constraints admits a polynomial-time $(\alpha, (\alpha + \triangle) \cdot b(v))$-approximation algorithm. For the case of unit weights, the* DWDCN *problem admits a polynomial-time $(\alpha, \alpha \cdot b(v) + \triangle - 1)$-approximation algorithm.*

Chapter 4

Approximation bounds for intersecting supermodular connectivity requirements

In this chapter, we analyse the approximation algorithms for the DWDCN problems with intersecting supermodular connectivity requirements, given in the previous chapter (Section 3.4) and derive new improved approximation bounds for these problems. In Section 4.1, we consider the DWDCN problem with weighted out-degree constraints. In Section 4.2, we consider the DWDCN problem with weighted in-degree constraints. In Section 4.3, we consider the DWDCN problem with both out- and in-degree constraints.

4.1 DWDCN with weighted out-degree constraints

In this section, we prove Theorem 3.1. Nutov [50, 52] showed that for intersecting supermodular f polytopes $P(f,b;\alpha,J,B')$ are $(2,5)$-sparse and $(3,3)$-sparse, and using Lemma 3.6 he proved that the algorithm described in Section 3.4 can give $(2,7b(v))$- and $(3,6b(v))$-approximation. We will show that polytopes $P(f,b;\alpha,J,B')$ are actually $(2,4)$-sparse (Lemma 4.2) and $(3,2)$-sparse (Lemma 4.1), which improve the results of [50, 52]. We further improve this to $(2,3)$-sparseness (Lemma 4.3).

Lemma 4.1. *For any $J \subseteq E$ and $B' \subseteq B$, polytope $P(f,b;\alpha,J,B')$ is $(2,4)$-sparse for intersecting supermodular f.*

Lemma 4.2. *For any $J \subseteq E$ and $B' \subseteq B$, polytope $P(f,b;\alpha,J,B')$ is $(3,2)$-sparse for intersecting supermodular f.*

Lemma 4.3. *For any $J \subseteq E$ and $B' \subseteq B$, polytope $P(f,b;\alpha,J,B')$ is $(2,3)$-sparse for intersecting supermodular f.*

Theorem 3.1 follows from Lemma 3.6 and Lemma 4.3. Hence, to complete the proof of Theorem 3.1, it only remains to prove Lemma 4.3, which will be given in Section 4.1.4. The reason for giving Lemmas 4.1 and 4.2 is that they are used in the proof of Theorem 3.4, which deals with the DWDCN problem with (both) weighted in- and out-degree constraints. The proofs of Lemmas 4.1 and 4.2 are given in Sections 4.1.2 and 4.1.3.

4.1.1 The structure of basic solutions and the outline of the sparseness proof

If $x \in P(f,b;\alpha,J,B')$ is a basic solution such that $0 < x(e) < 1$ for all $e \in E$, then each tight constraint (for x) is either

- a tight cut constraint $x(\delta_E^{in}(S)) = f_J(S)$ defined by some set $\emptyset \neq S \subsetneq V$ with $f_J(S) \geq 1$, or
- a tight weighted degree constraint $\sum_{e \in \delta_E^{out}(v)} x(e)w(e) = b_{\alpha,J}(v)$ defined by some node $v \in B'$.

The following lemma was proven in [50].

Lemma 4.4. *[50] For any basic solution $P(f,b;\alpha,J,B')$ with $0 < x(e) < 1$ for all $e \in E$, there exists a laminar family $\mathscr{L}$ on V and $T \subseteq B'$ such that $f_J(S) \geq 1$ for all $S \in \mathscr{L}$, and such that $x \in P$ is the unique solution to the system of linear equations:*

$$x(\delta_E^{in}(S)) = f_J(S), \text{ for all } S \in \mathscr{L}, \tag{4.1}$$

$$\sum_{e \in \delta_E^{out}(v)} x(e)w(e) = b_{\alpha,J}(v), \text{ for all } v \in T, \tag{4.2}$$

and $|\mathscr{L}| + |T| = |E|$. In particular, the characteristic vectors of $\{\delta_E^{in}(S) : S \in \mathscr{L}\}$ are linearly independent.

For a basic solution $P(f,b;\alpha,J,B')$, and $\mathscr{L}$ and T as in the lemma above, we define a child-parent relation on the members of $\mathscr{L} \cup T$ as follows. For $S \in \mathscr{L}$ or $v \in T$, its parent is the inclusion minimal member of $\mathscr{L}$ properly containing it, if any. Note that when $v \in T$ and $\{v\} \in \mathscr{L}$, then $\{v\}$ is the parent of v, and that no member of T has a child. This relation defines a forest F with the "vertex" set $\mathscr{L} \cup T$. Given $S \in \mathscr{L}$, we define C_S as the union of the sets of edges $\delta_E^{in}(R)$ for all children R of S in $\mathscr{L}$, that is, $\bigcup\{\delta_E^{in}(R) : R \text{ is a child in } \mathscr{L} \text{ of } S\}$. The following lemma is proven in [50, 52].

Lemma 4.5. *[50, 52] At least one of the sets $\delta_E^{in}(S) \setminus C_S$ and $C_S \setminus \delta_E^{in}(S)$ must be non-empty. If one of the sets $\delta_E^{in}(S) \setminus C_S, C_S \setminus \delta_E^{in}(S)$ is empty, then the other has more than α edges (hence, at least $\alpha + 1$, if α is integral).*

The previous sparseness proofs and our proofs of Lemmas 4.1 – 4.3 are based on assigning tokens to the edges of the graph G and then redistributing these tokens throughout the forest F. The total number of tokens in the graph is $2|E|$. We first perform an *initial assignment* of these $2|E|$ tokens, which satisfies the following property. Each node $v \in T$ gets 2 tokens (referred to as node-tokens), which are fixed at v and never used during the reassignment process. The remaining $2|E| - 2|T|$ tokens are distributed, in some way, among the edges and designated as head- or tail-tokens. Both head- and tail-tokens may be fractional.

Definition 2. *The head-token of an edge $(u,v) \in E$ is an S-token, if $v \in S$. The tail-token of an edge (u,v) is an S-token, if and only if, both $u,v \in S$. A node-token of $v \in V$ is an S-token, if and only if, $v \in S$.*

This definition implies that for a set $S \in \mathscr{L}$, if $R \in \mathscr{L}$ is a child of S in the forest, then all R-tokens are also S-tokens. It also implies that the tail-token of edge (u,v) leaving S (i.e, $u \in S, v \in V \setminus S$) is not an S-token.

Based on the initial assignment and using Definition 2, we will redistribute the $2|E|$ tokens throughout the forest F in a way that each proper descendant of S in forest F gets 2 S-tokens and S gets $2+z$ S-tokens with $z > 0$. We say that a $(2,2+z)$*-token reassignment is feasible*, if we can redistribute the $2|E|$ tokens in such a manner. A formal definition of the $(2,2+z)$-token reassignment is given below.

Definition 3. *A $(2,2+z)$-token reassignment scheme is an initial assignment of $2|E|$ tokens as defined in the paragraph before Definition 2 and a sequence of assignments of the tokens to (some of) the vertices of the forest F, with one assignment for each $S \in \mathscr{L}$. The assignment for $S \in \mathscr{L}$ assigns S-tokens to vertices of the subtree of F rooted at S in such a way that each descendant of S gets 2 S-tokens, and S gets $2+z$ S-tokens, where $z > 0$.*

We remark that similar notions of the token reassignment scheme are used in [50, 52, 38, 35, 62]. However, in these approaches, the parameter z is set to integral. By generalising this token reassignment scheme so that the parameter z may take on possibly fractional (positive) values and by finding more efficient initial assignment of the tokens, we obtain stronger sparseness properties.

The proofs of Lemmas 4.1 – 4.3 are by contradiction, so for convenience we state the *Negation Assumption* for the condition of $(\alpha, \triangle)$-sparseness. A polytope $P(f,b;\alpha,J,B')$ is not $(\alpha,\triangle)$-sparse, if and only if:

1. $|\delta_E^{out}(v)| \geq \triangle + 1$, for every $v \in B'$, and

2. there exists a basic solution $x \in P(f,b;\alpha,J,B')$ such that $0 < x(e) < 1/\alpha$, for every $e \in E$.

Note that the second condition implies that $|\delta_E^{in}(S)| \geq \alpha + 1$ for each $S \in \mathscr{L}$, since $x(\delta_E^{in}(S)) = \sum_{e \in \delta_E^{in}(S)} x(e) = f_J(S)$ is a positive integer.

The Negation Assumption for $(\alpha, \triangle)$ sparseness, the definition of the forest F and the feasibility of $(2, 2+z)$-token reassignment scheme give us a contradiction to Lemma 4.4 that $|E| = |\mathscr{L}| + |T|$. This implies that the polytope is $(\alpha, \triangle)$-sparse. Further details will be given in each sparseness proof.

4.1.2 $(2,4)$-sparseness of $P(f,b;\alpha,J,B')$ - Proof of Lemma 4.1

We assume the Negation Assumption for $(2,4)$-sparseness and take a basic solution x which satisfies Condition 2 of this assumption. For this basic solution $x \in P(f,b;\alpha,J,B')$, let $\mathscr{L}$ be a laminar family and T be a set of nodes as defined in Lemma 4.4. Recall that the constraints corresponding to the sets in $\mathscr{L}$ and to the nodes in T are tight for x and $|\mathscr{L}| + |T| = |E|$. The Negation Assumption implies that we have $|\delta_E^{in}(S)| \geq 3$ for all $S \in \mathscr{L}$ (from Condition 2 of the Negation Assumption) and $|\delta_E^{out}(v)| \geq 5$ for all $v \in T$ (from Condition 1 of the Negation Assumption).

We first perform an initial assignment of $2|E|$ tokens as follows.

Initial assignment

The total number of tokens in graph G is $2|E|$. For each edge $e = (v,u) \in E$ such that $v \notin T$, we designate 1 token as the head-token of e and 1 token as the tail-token of e. For each $v \in T$, there are $|\delta_E^{out}(v)| \geq 5$ outgoing edges from v. We designate 2 tokens as the node-tokens of v and 1 token as the head-token of each edge $e \in \delta_E^{out}(v)$. We also designate tail-tokens for the edges in $\delta_E^{out}(v)$ according to the following two cases. If v has exactly one $\mathscr{L}$-sibling R in F and there is an edge (v,x) such that $x \in R$, then we designate 1 token as the tail-token of this edge and $1/2$

token as the tail-tokens of the remaining edges in $\delta_E^{out}(v)$. Otherwise, we designate 1/2 token as the tail-token of each edge $e \in \delta_E^{out}(v)$. In both cases, since $|\delta_E^{out}(v)| \leq 5$, the total number of the tail-tokens assigned to edges in $\delta_E^{out}(v)$ is at most $|\delta_E^{out}(v)| - 2$. Thus, the total number of tokens used in the initial assignment is at most

$$\begin{aligned} &\sum_{v \notin T} \sum_{(v,u) \in E} (1+1) + \sum_{v \in T} (2 + |\delta_E^{out}(v)| + |\delta_E^{out}(v)| - 2) \\ &= \sum_{v \notin T} 2\big|\delta_E^{out}(v)\big| + \sum_{v \in T} 2\big|\delta_E^{out}(v)\big| \;=\; 2|E|. \end{aligned}$$

Note that the 2 node-tokens are fixed at $v \in T$ and never used during the reassignment process. This implies that node $v \in T$, which does not have a parent in F also gets 2 tokens. We also note that the initial assignment in Nutov [52] puts 1/2 tail-token for each edge (v,x) with $v \in T$. We obtain a stronger sparseness property by considering the special edges which can be given 1 tail-token.

Using the initial assignment above, we will show in Claim 1 that a $(2,3)$-token reassignment is feasible (see Definition 3). We obtain a contradiction as follows. Let the *root tokens* refer to the S-tokens for all $S \in \mathscr{L}$, which are roots of trees in the forest F. Let $\hat{T} \subseteq T$ be the set of nodes in T, which correspond to single-vertex trees in F and let $\tilde{T} = T \setminus \hat{T}$ be the set of nodes in T, which have a parent in F. The number of the root tokens in F is at most $2|E| - (2+5/2)|\hat{T}|$ since the two node-token and 1/2 tail-tokens of each $v \in \hat{T}$ are not S-tokens for any $S \in \mathscr{L}$. The feasibility of a $(2,3)$-token reassignment scheme implies that there are at least $2(|\mathscr{L}| + |\tilde{T}|) + 1$ root-tokens. Hence $2|E| - (2+5/2)|\hat{T}| \geq 2(|\mathscr{L}| + |\tilde{T}|) + 1$, so we have $2|E| \geq 2(|\mathscr{L}| + |\tilde{T}| + |\hat{T}|) + 1 + 5/2|\hat{T}| > 2(|\mathscr{L}| + |T|)$. This contradicts $|E| = |\mathscr{L}| + |T|$, implying that polytope $P(f,b;\alpha,J,B')$ is $(2,4)$-sparse.

The remaining part of the proof of Lemma 4.1 is the proof of the following claim.

Claim 1. *A $(2,3)$-token reassignment scheme is feasible if polytope $P(f,b;\alpha,J,B')$ is not $(2,4)$-sparse.*

Proof. Based on the initial assignment above, we show that a $(2,3)$-token reassignment scheme is feasible. The proof is given by induction on the number of $\mathscr{L}$-descendants of S in F. Recall that each node $v \in T$ has two node-tokens, which are fixed at node v and never used during the reassignment process. By Definition 2, all these node-tokens are S-tokens for each $S \in \mathscr{L}$ such that $v \in S$. Therefore, $v \in T$ always gets the required number of S-tokens if v is a descendant of S in F.

Base case: S has no $\mathscr{L}$-descendants in F. We assign to S the head-tokens from any three edges $e \in \delta_E^{in}(S)$. This is feasible since for each $S \in \mathscr{L}$, $|\delta_E^{in}(S)| \geq 3$. By Definition 2, all these head-tokens (3 tokens) are S-tokens. Hence S gets an assignment as required.

Inductive steps. Take S in the forest F which has at least one child R in $\mathscr{L}$. By the inductive hypothesis, we assume an assignment of R-tokens to the subtree rooted in R for each child R of S. Each child $R \in \mathscr{L}$ of S has 3 R-tokens and each descendant of R has 2 R-tokens. We consider three cases.

S **has at least 3 children in** $\mathscr{L}$**:** By moving 1 R-token from each child R to S, S gets at least 3 R-tokens. Each child R now has 2 remaining R-tokens. Hence, we get an assignment as required since by the definition, all R-tokens here are S-tokens as well.

S **has exactly 2 children, say** R_1 **and** R_2**, in** $\mathscr{L}$**:** By moving 1 R-tokens from each child to S, S gets 2 R-tokens, which are also S-tokens (see Definition 2), so S needs 1 more S-token.

If there is $(u,x) \in \delta_E^{in}(S) \backslash C_s$, then we assign to S the head-token of this edge. By Definition 2, this head-token (1 token) is an S-token. Hence, S gets an assignment as required.

If there is no such edge, i.e, $\delta_E^{in}(S) \backslash C_S = \emptyset$, then $C_S \backslash \delta_E^{in}(S)$ has at least $\alpha + 1 = 3$ edges (see Lemma 4.5). Among these edges, if there is $(v,u) \in C_S \backslash \delta_E^{in}(S)$ such that $v \notin T$, then we assign to S the tail-token (1 token) of this edge. By Definition 2, this tail-token is an S-token. Hence, S gets an assignment as required.

Otherwise, for all $(v,u) \in C_S \backslash \delta_E^{in}(S), v \in T$. Recall that during the initial assignment, we designated $1/2$ token as the tail-token of each edge $(v,u) \in \delta_E^{out}(v)$, if $v \in T$. We assign to S all tail-tokens from $(v,u) \in C_S \backslash \delta_E^{in}(S)$. This way S gets at least 3 tail-tokens since

$|C_S \setminus \delta_E^{in}(S)| \geq 3$. Note that these tail-tokens (3/2 tokens), which we assigned to S, are from either (i) $(v,u) \in \delta_E^{out}(v)$ such that $v \in R$ and $u \in R'$, where R is a child of S and R' is a sibling of R, or (ii) $(v,u) \in \delta_E^{out}(v)$ such that v is a child of S and has at least two siblings in $\mathscr{L}$. By Definition 2, the tail-tokens from (i) are S-tokens and they are not R-tokens for a descendant R of S. This is because u is not in R. Therefore these tail-tokens have not been assigned before. The tail-tokens from (ii) are clearly S-tokens and not R-tokens. Thus, S gets the required number of S-tokens.

S has exactly 1 child R in $\mathscr{L}$: S gets 1 R-token from R (which is also S-token), so it needs 2 more S-tokens. Lemma 4.5 implies that we have only the following sub-cases: (i) both sets $C_S \setminus \delta_E^{in}(S)$ and $\delta_E^{in}(S) \setminus C_S$ have at least 1 edge, (ii) $C_S \setminus \delta_E^{in}(S)$ is empty and $\delta_E^{in}(S) \setminus C_S$ has at least 3 edges, (iii) $\delta_E^{in}(S) \setminus C_S$ is empty and $C_S \setminus \delta_E^{in}(S)$ has at least 3 edges.

(i) Both sets $C_S \setminus \delta_E^{in}(S)$ and $\delta_E^{in}(S) \setminus C_S$ have at least 1 edge.

We assign to S the head-token from edge $e \in \delta_E^{in}(S) \setminus C_S$. This head-token (1-token) is an S-token by Definition 2. Hence, S gets 1 S-token, so S needs 1 more S-token. We assign to S the tail-token from $(v,z) \in C_S \setminus \delta_E^{in}(S)$. Recall that during the initial assignment if there exists an edge $(v,z) \in \delta_E^{out}(v)$ such that $v \in T$ and $z \in R$, then we designate 1 token as the tail-token of this edge. Therefore, in any case, either $v \in T$ or $v \notin T$, S gets one tail-token (1 token) from $(v,z) \in C_S \setminus \delta_E^{in}(S)$. By Definition 2, this tail-token is an S-token. Therefore, S gets the required number of S-tokens.

(ii) $C_S \setminus \delta_E^{in}(S)$ is empty and $\delta_E^{in}(S) \setminus C_S$ has at least 3 edges.

We assign to S the head-tokens from any two edges $e \in \delta_E^{in}(S) \setminus C_S$. Since these head-tokens (2 tokens) are S-tokens by Definition 2, S gets the required number of S-tokens.

(iii) $\delta_E^{in}(S) \setminus C_S$ is empty and $C_S \setminus \delta_E^{in}(S)$ has at least 3 edges.

We assign to S the tail-tokens from all edges $(v,u) \in C_S \setminus \delta_E^{in}(S)$. Again, the tails of these edges may or may not be in T. If none of them is in T, then S gets at least three tail-tokens (3 tokens) since $|C_S \setminus \delta_E^{in}(S)| \geq 3$. Hence, S gets an assignment as required. Now consider the tight case that none of edges $(v,z) \in C_S \setminus \delta_E^{in}(S)$ are such that $v \notin T$, i.e, all edges $(v,z) \in C_S \setminus \delta_E^{in}(S)$ are such that $v \in T$. Recall that during the initial assignment, if there exists $(v,z) \in \delta_E^{out}(v)$ such that $z \in R$ and $v \in T$, then we designated 1 token as

the tail-token of this edge, and we designated 1/2 token as the tail-token of the remaining edges in $\delta_E^{out}(v)$. Therefore, S gets at least 3 tail-tokens (2 tokens). By Definition 2, all these tail-tokens (2 tokens) are S-tokens, S gets an assignment as required.

□

4.1.3 $(3,2)$-sparseness of $P(f,b;\alpha,J,B')$ - Proof of Lemma 4.2

We prove that $P(f,b;\alpha,J,B')$ is $(3,2)$-sparse by contradiction, assuming the Negation Assumption for $(3,2)$-sparseness and taking a basic solution x satisfying Condition 2 of this assumption. For this basic solution $x \in P(f,b;\alpha,J,B')$, let $\mathscr{L}$ be a laminar family and $T \subseteq B'$ be a set of nodes as defined in Lemma 4.4. Recall that the constraints corresponding to the sets in $\mathscr{L}$ and to the nodes in T are tight for x and $|\mathscr{L}|+|T| = |E|$. Thus, we have $|\delta_E^{in}(S)| \geq 4$ for all $S \in \mathscr{L}$ (from Condition 2 of the Negation Assumption) and $|\delta_E^{out}(v)| \geq 3$ for all $v \in T$ (from Condition 1 of the Negation Assumption).

We first perform an initial assignment of $2|E|$ tokens given below. We note that in addition to 2 fixed node-tokens at the nodes in T we now also initially have *spare* node-tokens. We use the same definition of S-tokens as in Definition 2.

Initial assignment

The total number of tokens in graph G is $2|E|$. For each edge $(v,u) \in E$ such that $v \notin T$, we designate 1 token as the head-token and 1 token as the tail-token. For each $v \in T$, there are $|\delta_E^{out}(v)| \geq 3$ outgoing edges from node v. We designate $\delta_E^{out}(v)$ tokens as the node-tokens of v and 1 token as the head-token of each $e \in \delta_E^{out}(v)$. Hence each node $v \in T$ has at least 3 node-tokens. Among these node-tokens, we fix 2 tokens for v. Therefore each node $v \in T$ has at least 1 *spare* node-token available. The spare node-tokens will be used during the reassignment.

The fixed two tokens are never assigned during reassignment process. The total number of tokens used in this initial assignment is at most

$$\sum_{v\notin T}\sum_{(v,u)\in E}(1+1)+\sum_{v\in T}\left(\sum_{(v,u)\in E}(1)\right)+\left|\delta_E^{out}(v)\right|$$
$$=\sum_{v\notin T}2\left|\delta_E^{out}(v)\right|+\sum_{v\in T}2\left|\delta_E^{out}(v)\right|=2|E|.$$

Using this initial assignment, we will show later in Claim 2 that a $(2,4)$-token reassignment is feasible (see Definition 3).

We obtain a contradiction as follows. Let the *root tokens* refer to the S-tokens for all $S\in\mathscr{L}$, which are roots of trees in the forest F. Let $\hat{T}\subseteq T$ be the set of nodes in T, which correspond to single-vertex trees in F, and let $\tilde{T}=T\setminus\hat{T}$ be the set of nodes in T, which have a parent in F. The number of the root tokens in F is at most $2|E|-3|\hat{T}|$ since the three node-tokens of each $v\in\hat{T}$ are not S-tokens for any $S\in\mathscr{L}$. The feasibility of a $(2,4)$-token reassignment scheme implies that there are at least $2(|\mathscr{L}|+|\tilde{T}|)+2$ root tokens. Hence, $2|E|-3|\hat{T}|\geq 2(|\mathscr{L}|+|\tilde{T}|)+2$, so we have $2|E|\geq 2(|\mathscr{L}|+|\tilde{T}|+|\hat{T}|)+2+|\hat{T}|>2\,(|\mathscr{L}|+|T|)$. This contradicts $|E|=|\mathscr{L}|+|T|$ implying that $P(f,b;\alpha,J,B')$ is $(3,2)$-sparse.

The remaining part of the proof of Lemma 4.2 is the proof of the following Claim 2.

Claim 2. *A $(2,4)$-token reassignment scheme is feasible if polytope $P(f,b;\alpha,J,B')$ is not $(3,2)$-sparse..*

Proof. Based on the initial assignment above, we show that a $(2,4)$-token reassignment scheme is feasible. The proof is given by induction on the number of $\mathscr{L}$-descendants of S in F. Recall that for each node $v\in T$, we have at least three nodes-tokens. Among these node-tokens, two node-tokens are fixed for itself and hence each $v\in T$ has at least 1 spare node-token available (this token will be used for other assignment). By Definition 2, the fixed node-tokens at v are S-tokens if v is in S. Therefore, $v\in T$ always gets the required number of S-tokens if v is a descendant of S in F.

Base case: S has no $\mathscr{L}$-descendants in F. We assign to S the head-tokens from any four edges $(u,v) \in \delta_E^{in}(S)$. This is feasible since for each $S \in \mathscr{L}$, $|\delta_E^{in}(S)| \geq 4$. By definition, all these head-tokens (4 tokens) are S-tokens. Hence S gets an assignment as required.

Inductive steps. Take S in the forest F which has at least one child R in $\mathscr{L}$. By the inductive hypothesis, we assume an assignment of R-tokens to the subtree rooted in R for each child R of S. Each child $R \in \mathscr{L}$ of S has 4 R-tokens and each descendant of R has 2 R-tokens. We consider two cases.

S has at least 2 children in $\mathscr{L}$: By moving 2 R-tokens from each child R to S, S gets at least 4 R-tokens. Each child R now has 2 remaining R-tokens. Hence, we get an assignment as required since by the definition, all R-tokens are S-tokens as well.

S has exactly 1 child in $\mathscr{L}$: By moving 2 R-tokens from R, S gets 2 R-tokens (also S-tokens), but needs 2 more S-tokens. Lemma 4.5 implies that we have only the following sub-cases: (i) both sets $C_S \setminus \delta_E^{in}(S)$ and $\delta_E^{in}(S) \setminus C_S$ have at least 1 edge, (ii) $C_S \setminus \delta_E^{in}(S)$ is empty and $\delta_E^{in}(S) \setminus C_S$ has at least 4 edges, (iii) $\delta_E^{in}(S) \setminus C_S$ is empty and $C_S \setminus \delta_E^{in}(S)$ has at least 4 edges.

(i) Both sets $C_S \setminus \delta_E^{in}(S)$ and $\delta_E^{in}(S) \setminus C_S$ have at least 1 edge.

We assign to S the head-token from edge $e \in \delta_E^{in}(S) \setminus C_S$. Hence, S gets 1 head-token (S-token), so S needs 1 more S-token. Now consider edge in $(v,u) \in C_S \setminus \delta_E^{in}(S)$. If $v \notin T$, then we assign to S the tail-token from $(v,u) \in C_S \setminus \delta_E^{in}(S)$. This tail-token is an S-token by Definition 2. If $v \in T$, then we assign to S one node-token (S-token) from v (Recall that each $v \in T$ has at least 1 spare node-token available). Therefore, in any case, either $v \in T$ or $v \notin T$, S gets 1 S-tokens. Hence, S gets in total 2 S-tokens, thus S gets the required number of S-tokens.

(ii) $C_S \setminus \delta_E^{in}(S)$ is empty and $\delta_E^{in}(S) \setminus C_S$ has at least 4 edges.

We assign to S the head-tokens from any two edges $(u,v) \in \delta_E^{in}(S) \setminus C_S$. All these head-tokens (2 tokens) are S-tokens by Definition 2. Hence, S gets the required number of S-tokens.

(iii) $\delta_E^{in}(S) \setminus C_S$ is empty and $C_S \setminus \delta_E^{in}(S)$ has at least 4 edges.

If S does not have a child $v \in T$, then we assign to S all the tail-tokens from $(v,z) \in C_S \setminus \delta_E^{in}(S)$. This way S gets at least 4 tail-tokens. All these tail-tokens (4 tokens) are by Definition 2 S-tokens. Thus, S gets the required number of S-tokens.

Now assume that S has a child v in T. We consider separately the cases when $|C_S \setminus \delta_E^{in}(S) \cap \delta_E^{out}(v)| \geq 4$ and when $|C_S \setminus \delta_E^{in}(S) \cap \delta_E^{out}(v)| \leq 3$. First consider the case when $|C_S \setminus \delta_E^{in}(S) \cap \delta_E^{out}(v)| \geq 4$. This implies that $|\delta_E^{out}(v)| \geq 4$. By the initial assignment, we know that v has 2 fixed node-tokens and at least 2 spare node-tokens. By moving these 2 spare node-tokens (S-tokens) to S, S gets the required number of S-tokens. Now consider the other case that $|C_S \setminus \delta_E^{in}(S) \cap \delta_E^{out}(v)| \leq 3$. Since $|C_S \setminus \delta_E^{in}(S)| \geq 4$, there exists a node y in S such that $(y,u) \in C_S \setminus \delta_E^{in}(S)$. If $y \notin T$, then we assign to S the tail-token of this edge and the spare 1 token from v. Hence, S gets 2 tokens, which by Definition 2, are S-tokens. If $y \in T$, we know that y also has at least 1 spare node-token available. Therefore, by moving the spare node-token from v and y, S gets 2 additional S-tokens. Thus S gets the required number of S-tokens.

□

4.1.4 $(2,3)$-sparseness of $P(f,b;\alpha,J,B')$ - Proof of Lemma 4.3

We assume the Negation Assumption for $(2,3)$-sparseness (see page 31) and take a basic solution x satisfying Condition 2 of this assumption. For this basic solution $x \in P(f,b;\alpha,J,B')$, let $\mathscr{L}$ be a laminar family and T be a set of nodes as defined in Lemma 4.4. Hence, we have $|\delta_E^{in}(S)| \geq 3$ for all $S \in \mathscr{L}$ and $|\delta_E^{out}(v)| \geq 4$ for all $v \in T$.

Similarly as in the proofs of Lemmas 4.1 and 4.2, we initially have the total number of tokens is $2|E|$. In the proof of Lemma 4.2, these tokens are redistributed to the edges (heads and tails) and nodes of graph. At the time of the initial assignment, head and tail-tokens are discrete unit tokens and remain such during the reassignment process. Based on this initial assignment, we inductively show that a $(2,4)$-token reassignment scheme is feasible, if polytope

is not $(3,2)$-sparse. This implies that the polytope must be $(3,2)$-sparse since any $(2,4)$-token reassignment scheme needs more than $2|E|$ tokens. In the proof of Lemma 4.1, head-tokens remain discrete unit tokens but tail-tokens are split into some fractions and are reassigned to the edges and nodes during the initial assignment. Again, based on the initial assignment, we inductively show that a $(2,3)$-token reassignment scheme is feasible, if polytope is not $(2,4)$-sparse. Now, in the proof of Lemma 4.3, we allow both head and tail-tokens to be fractional. Thereby we find a more efficient initial assignment which leads to a $(2,2+z)$-token reassignment scheme for some $0 < z < 1$, if the polytope is not $(2,3)$-sparse.

We first prove the following claim and then we will complete the proof of Lemma 4.3.

Claim 3. *If polytope $P(f,b;\alpha,J,B')$ is not $(2,3)$-sparse, then there exists $0 < z < 1$ such that a $(2,2+z)$-token reassignment scheme is feasible with $z > 0$.*

Proof. In addition to the parameter z, we use in the proof also non-negative real parameters h, t_1, t_2. The values of all these parameters will be set at the end of the proof. Before we redistribute the tokens inductively using the forest structure F of the laminar family $\mathscr{L}$, we have to specify the initial assignment of tokens.

Initial assignment

Recall that the total number of tokens in the graph is $2|E|$. For each $e = (v,u) \in E$ such that $v \notin T$, we designate h tokens as head-tokens of e and $2-h$ tokens as tail-tokens of e, for some real $0 \leq h \leq 2$. For each $v \in T$, there are $|\delta_E^{out}(v)| \geq 4$ outgoing edges from node v. We designate 2 tokens as node-tokens of v and h tokens as head-tokens of each edge $e \in \delta_E^{out}(v)$. We also designate tail-tokens for the edges in $\delta_E^{out}(v)$ according to the following two cases. If v has exactly one $\mathscr{L}$-sibling R in forest F and there is at least one edge (v,x) such that $x \in R$, then we choose one of these edges, designate t_1 tokens as tail-tokens of this edge and designate t_2 tokens as tail-tokens for each of the other edges in $\delta_E^{out}(v)$. Otherwise, we designate t_2 tokens as tail-tokens for each edge (v,u). The reason for these two cases will be clear when we show the inductive reassignment later in the proof. To ensure that we do not exceed the total of $2|E|$

tokens, we require that h, t_1, and t_2 satisfy the following constraints.

$$2+|\delta_E^{out}(v)|\cdot h+t_1+(|\delta_E^{out}(v)|-1)\cdot t_2 \leq 2|\delta_E^{out}(v)|, \tag{4.3}$$

$$2+|\delta_E^{out}(v)|\cdot h+|\delta_E^{out}(v)|\cdot t_2 \leq 2|\delta_E^{out}(v)|. \tag{4.4}$$

Thus the number of tokens used in the initial assignment is at most $\sum_{v\notin T}\sum_{(v,u)\in E}((h+(2-h))+\sum_{v\in T}2|\delta_E^{out}(v)| = 2|E|$.

Constraint (4.4) implies that $h+t_2<2$ and this constraint is clearly monotone with respect to the value of $|\delta_E^{out}(v)|$ (that is, if (4.4) is true for $|\delta_E^{out}(v)|=4$, then it is also true for any $|\delta_E^{out}(v)|\geq 4$). Constraint (4.3) can be rearrange as $2+t_1-t_2+|\delta_E^{out}(v)|\cdot(h+t_2)\leq 2|\delta_E^{out}(v)|$, hence it is also monotone with respect to $|\delta_E^{out}(v)|$ (observe that $2+t_1-t_2\geq 0$, because $t_2<2$). Therefore, to ensure that (4.3) and (4.4) hold whenever $|\delta_E^{out}(v)|\geq 4$, we require

$$2+4h+t_1+3t_2 \leq 8, \text{ and} \tag{4.5}$$

$$2+4h+4t_2 \leq 8. \tag{4.6}$$

We remark that the node-tokens are fixed in the sense that they will never be reassigned anywhere else during the inductive steps. Therefore node $v\in T$ always gets the required number of S-tokens, if v is a descendant of S in F.

Now using this initial assignment we show that a $(2,2+z)$-token reassignment scheme is feasible. The proof is by induction on the number of $\mathscr{L}$-descendants of S in F.

Base case: S has no $\mathscr{L}$-descendants in F.

We assign to S the head-tokens from all edges $(u,v)\in\delta_E^{in}(S)$. All these tokens are S-tokens by definition. This way S gets at least $3h$ tokens since $\left|\delta_E^{in}(S)\right|\geq 3$. To make sure that S gets this way at least $2+z$ tokens we require

$$3h\geq 2+z. \tag{4.7}$$

Inductive steps. Take now S in the forest F which has at least one child R in $\mathscr{L}$. Recall the notation that C_S is the union of the sets of edges $\delta_E^{in}(R)$ for all children R of S in $\mathscr{L}$. By inductive hypothesis, we assume that for each child $R \in \mathscr{L}$ of S, R has at least $2+z$ R-tokens and each descendant of R has at least 2 R-tokens.

We consider separately the case when S has exactly one child R in $\mathscr{L}$ and the case when S has at least two children in $\mathscr{L}$. Lemma 4.5 implies that we have only the following sub-cases: (i) both sets $C_S \setminus \delta_E^{in}(S)$ and $\delta_E^{in}(S) \setminus C_S$ have at least 1 edge, (ii) $C_S \setminus \delta_E^{in}(S)$ is empty and $\delta_E^{in}(S) \setminus C_S$ has at least 3 edges, (iii) $\delta_E^{in}(S) \setminus C_S$ is empty and $C_S \setminus \delta_E^{in}(S)$ has at least 3 edges.

S has exactly 1 child R in $\mathscr{L}$: We move z tokens from R to S. Now R has at least 2 R-tokens and S has z R-tokens. Each descendant of R also has at least 2 R-tokens. By Definition 2, all these R-tokens are S-tokens. Thus R and its descendants have the required number of S-tokens, but S needs 2 additional S-tokens.

(i) Both sets $C_S \setminus \delta_E^{in}(S)$ and $\delta_E^{in}(S) \setminus C_S$ have at least 1 edge.
We assign to S the h head-tokens from one edge $e \in \delta_E^{in}(S) \setminus C_S$ and the tail-tokens from one edge $(v,u) \in C_S \setminus \delta_E^{in}(S)$, which are by definition S-tokens (observe that $v \in S$). Note that node v may or may not be in T. If $v \notin T$, then edge (v,u) has $2-h$ tail-tokens and if $v \in T$, then edge (v,u) has t_1 tail-tokens, Hence, to guarantee that S gets at least 2 additional tokens, we require

$$h + t_1 \geq 2. \tag{4.8}$$

(ii) $C_S \setminus \delta_E^{in}(S)$ is empty and $\delta_E^{in}(S) \setminus C_S$ has at least 3 edges.
We assign to S the head-tokens from all edges $e \in \delta_E^{in}(S) \setminus C_S$. This way S gets at least $3h$ tokens, which are by definition S-tokens. The constraints (4.7) implies that S gets at least 2 additional S-tokens.

(iii) $\delta_E^{in}(S) \setminus C_S$ is empty and $C_S \setminus \delta_E^{in}(S)$ has at least 3 edges.
We assign to S the tail-tokens from all edges $(v,u) \in C_S \setminus \delta_E^{in}(S)$. Again, the tails of these edges may or may not be in T. If none of them is in T, then S gets $3(2-h)$ tail-tokens. If $(v,u) \in C_S \setminus \delta_E^{in}(S)$ and $v \in T$, then S gets t_1 tail-tokens from this edge and at least

$\min\{2-h, t_1, t_2\}$ tail-tokens from each of the remaining edges in $C_S \setminus \delta_E^{in}(S)$. Therefore, we require

$$3(2-h) \geq 2, \tag{4.9}$$

$$t_1 + 2 \cdot \min\{2-h, t_1, t_2\} \geq 2. \tag{4.10}$$

S has at least 2 children in $\mathscr{L}$: For each child $R \in \mathscr{L}$ of S, we move z tokens from R to S. Each child R now has at least 2 R-tokens and S has at least $2z$ R-tokens. Each descendant of R also has at least 2 R-tokens. By definition, all these R-tokens are S-tokens. Thus R and its descendants have the required number of S-tokens, but S needs $2-z$ additional S-tokens.

(i) Both sets $C_S \setminus \delta_E^{in}(S)$ and $\delta_E^{in}(S) \setminus C_S$ have at least 1 edge.
We assign to S the h head-tokens from one edge $e \in \delta_E^{in}(S) \setminus C_S$ and the tail-tokens from one edge $(v,u) \in C_S \setminus \delta_E^{in}(S)$, which are by definition S-tokens. Again, node v may or may not be in T. If $v \notin T$, then edge (v,u) has $2-h$ tail-tokens and if $v \in T$, then edge (v,u) has t_2 tail-tokens. Therefore, to give S at least $2-z$ additional S-tokens we require that

$$h + t_2 \geq 2 - z. \tag{4.11}$$

(ii) $C_S \setminus \delta_E^{in}(S)$ is empty and $\delta_E^{in}(S) \setminus C_S$ has at least 3 edges.
We assign to S the head-tokens from all edges $e \in \delta_E^{in}(S) \setminus C_S$. Hence, S gets at least $3h \geq 2$ head-tokens, which are S-tokens (see constraints (4.7)).

(iii) $\delta_E^{in}(S) \setminus C_S$ is empty and $C_S \setminus \delta_E^{in}(S)$ has at least 3 edges.
We assign to S the tail-tokens from all edges $(v,u) \in C_S \setminus \delta_E^{in}(S)$. The tail-tokens can be either from (a) $(v,u) \in C_S \setminus \delta_E^{in}(S)$ such that v is not in any child of S, or (b) $(v,u) \in C_S \setminus \delta_E^{in}(S)$ such that v is in child R of S and u is in another child R' of S. Clearly, the tail-tokens in the case (a) have not been assigned before since they are not R-tokens for any child R of S. The node v in the case (b) has been considered previously as v is inside R. However, in this case, the tail-tokens assigned to edge (v,u) have not been used as

by definition, they are not R-tokens for the child R of S (the tail-tokens from (x,y) are R-tokens only if both x,y are in R). By definition, all these tail-tokens are S-tokens and each edge in $C_S \setminus \delta_E^{in}(S)$ has either $2-h$ (if $v \notin T$) or t_2 (if $v \in T$) tail-tokens. Hence, we require that

$$3 \cdot \min\{2-h, t_2\} \;\geq\; 2-z. \tag{4.12}$$

The existence of a feasible solution to the constraints (4.5)–(4.12) with $0 \leq h \leq 2$, $t_1 \geq 0$, $t_2 \geq 0$ and $0 < z$ implies that a $(2, 2+z)$-token reassignment scheme is feasible. It is easy to check that the following solution satisfies all constraints: $h = 8/9$, $t_1 = 10/9$, $t_2 = 4/9$, and $z = 2/3$. We remark that it turns out that this is the unique solution for our constraints.

□

Now we complete the proof of Lemma 4.3 using Claim 3. We reach the contradiction as before. We perform the initial assignment of tokens as described in Claim 3 and consider $(2, 2+z)$-token reassignment scheme for all S which are roots in F.

Let the root tokens refer to the S-tokens for all $S \in \mathscr{L}$ which are roots of trees in the forest F. Let $\hat{T} \subseteq T$ be the set of nodes in T which correspond to single-vertex trees in F, and let $\tilde{T} = T \setminus \hat{T}$ be the set of nodes in T, which have a parent in F. Note that the two node-tokens and 4/9 tail-tokens of each $v \in \hat{T}$ are not S-tokens for any $S \in \mathscr{L}$. Therefore, the Negation Assumption for $(2,3)$-sparseness implies that the number of the root tokens in F is at most $2|E| - (2 + 4 \cdot 4/9)|\hat{T}|$. The feasibility of a $(2,3)$-token reassignment scheme implies that there are at least $2(|\mathscr{L}| + |\tilde{T}|) + 2/3$ root tokens. Hence, $2|E| - (2 + 16/9)|\hat{T}| \geq 2(|\mathscr{L}| + |\tilde{T}|) + 2/3$, so we have $2|E| \geq 2(|\mathscr{L}| + |\tilde{T}| + |\hat{T}|) + 2/3 + 16/9|\hat{T}| > 2(|\mathscr{L}| + |T|)$. This contradicts $|E| = |\mathscr{L}| + |T|$, implying that polytope $P(f, b; \alpha, J, B')$ is $(2,3)$-sparse.

4.2 DWDCN with weighted in-degree constraints

In this section, we prove Theorem 3.2. Let $P(f, b^{in})$ be the polytope defined by the following constraints.

$$\begin{aligned} x\left(\delta_E^{in}(S)\right) &\geq f(S), \quad \text{for all} \quad \emptyset \neq S \subsetneq V, \\ \sum_{e \in \delta_E^{in}(v)} x(e)w(e) &\leq b^{in}(v), \quad \text{for all} \quad v \in B^{in}, \\ 0 \leq x(e) &\leq 1, \quad \text{for all} \quad e \in E. \end{aligned}$$

Given $\alpha \geq 1$, an edge set $J \subseteq E$, a subset $B'' \subseteq B^{in}$, let polytope $P(f, b^{in}; \alpha, J, B'')$ be defined as:

$$\begin{aligned} x\left(\delta_E^{in}(S)\right) &\geq f_J(S) \equiv f(S) - |\delta_J^{in}(S)|, && \text{for all} \quad \emptyset \neq S \subsetneq V, \\ \sum_{e \in \delta_E^{in}(v)} x(e)w(e) &\leq b^{in}_{\alpha,J}(v) \equiv b^{in}(v) - w(\delta_J^{in}(v))/\alpha, && \text{for all} \quad v \in B'' \subseteq B^{in}, \\ 0 \leq x(e) &\leq 1, && \text{for all} \quad e \in E. \end{aligned}$$

Observe that $P(f, b^{in}; \alpha, \emptyset, B^{in})$ is polytope $P(f, b^{in})$.

The algorithm given in Figure 3.1 can be easily modified to solve the DWDCN problem with weighted in-degree constraints, by considering polytope $P(f, b^{in})$ in the beginning of the algorithm, and by finding an optimal basic feasible solution x to the LP problem of minimising $\sum_{e \in E} c(e) \cdot x(e)$ over the polytope $P(f, b^{in}; \alpha, \emptyset, B^{in})$ at each iteration of the while loop.

For the DWDCN problem with weighted in-degree constraints, we now use the following definition of sparseness of polytope.

Definition 4. *Polytope $P(f, b^{in}; \alpha, J, B'')$ is $(\alpha, \triangle)$-sparse, if and only if, there exists a node $v \in B''$ with $|\delta_E^{in}(v)| \leq \triangle$, or for any basic solution $x \in P(f, b^{in}; \alpha, J, B'')$, there exists $e \in E$ such that $x(e) = 0$ or $x(e) \geq 1/\alpha$.*

Note that if $P(f, b^{in}; \alpha, J, B'')$ is empty, then it is trivially $(\alpha, \triangle)$-sparse.

Lemma 4.6. *For each integer* $\triangle \geq 1$*, if* $\alpha = 1 + 1/\triangle$*, then polytope* $P(f, b^{in}; \alpha, J, B'')$ *is* $(\alpha, \triangle)$*-sparse.*

The proof of Lemma 4.6 gives the following corollary.

Corollary 4.7. *Polytope* $P(f, b^{in}; \alpha, J, B'')$ *is* $(3/2, 2)$*-sparse and* $(2, 1)$*-sparse.*

Corollary 4.7 and Lemma 3.6 imply that there is a polynomial-time $(2, 3b(v))$- and $(3/2, 7/2b(v))$-approximation algorithm for the DWDCN problem with in-degree constraints under the intersecting supermodular connectivity.

Let $H' = (V, F')$ be the output of the above algorithms. From the graph H', we find an inclusion minimal edge set $F \subset F'$ on V such that $H = (V, F)$ is f-connected. Now Theorem 3.2 easily follows from Claim 4 below.

Claim 4. *[52] Let* f *be an intersecting supermodular function on* V*, and let* $F \subseteq E$ *be an inclusion-wise minimal edge set on* V *such that* $H = (V, F)$ *is* f*-connected. Then,* $|\delta_F^{in}(v)| \leq f_{\max}$*, where* $f_{\max} = \max\{f(S) : S \subseteq V\}$.

Hence, to complete the proof of Theorem 3.2, it only remains to prove Lemma 4.6. This will be given in the following section.

4.2.1 $(\alpha, \triangle)$-sparseness of $P(f, b^{in}; \alpha, J, B'')$- Proof of Lemma 4.6

Nutov [52] showed that a polytope $P(f, b^{in}; \alpha, J, B'')$ is $(1, 3)$-sparse. He proved this using the token reassignment approach described in Section 4.1. Our sparseness proof (the proof of Lemma 4.6) is similar to this approach in a sense that we exploit the structure of the forest $F = \mathscr{L} \cup T$. However, instead of counting the number of tokens, which were assigned to the edges in the graph, we simply count a number of edges and we will obtain a contradiction.

Similarly to Lemma 4.4, for any basic solution $x \in P(f, b^{in}; \alpha, J, B'')$ with $0 < x(e) < 1$ for all $e \in E$, we have a laminar family $\mathscr{L}$ on V and $T^{in} \subseteq B''$ such that the constraints corresponding to $\mathscr{L}$ and T^{in} are tight for x and uniquely define x. Equations (4.2) are now the following weighted in-degree constraints.

$$\sum_{e \in \delta_E^{in}(v)} x(e)w(e) = b^{in}_{\alpha,J}(v), \text{ for all } v \in T^{in}.$$

Thus, we have

$$|\mathscr{L}| + \left|T^{in}\right| = |E|. \tag{4.13}$$

We use the notation T^{in} to make it clear that we consider here the weighted in-degrees.

Again, we prove that $P(f, b^{in}; \alpha, J, B'')$ is $(\alpha, \triangle)$-sparse by contradiction assuming the following Negation Assumption for $(\alpha, \triangle)$-sparseness of the polytope $P(f, b^{in}; \alpha, J, B'')$:

1. $|\delta_E^{in}(v)| \geq \triangle + 1$, for every $v \in B''$, and
2. there exists a basic solution $x \in P(f, b^{in}; \alpha, J, B'')$ such that $0 < x(e) < 1/\alpha$, for every $e \in E$.

We take a basic solution x satisfying the second condition of the above negation assumption. For this basic solution $x \in P(f, b^{in}; \alpha, J, B'')$, let $\mathscr{L}$ be a laminar family and T^{in} be a set of nodes as defined above. For $\mathscr{L}$ and T^{in}, we consider the same forest F as defined for the proofs of Lemmas 4.2, 4.1 and 4.3. Further, for any $S' \in \mathscr{L}$, we define $D(S')$ as the number of vertices of the subtree of F rooted at S'. Let $\widehat{T^{in}} \subseteq T^{in}$ be the set of nodes in T^{in}, which correspond to single-vertex trees in F and let $\widetilde{T^{in}} = T^{in} \setminus \widehat{T^{in}}$ be the set of nodes in T^{in}, which have a parent in F. By the Negation Assumption, we have $|\delta_E^{in}(v)| \geq \triangle + 1$, for each node $v \in T^{in}$. Therefore we also have $|\delta_E^{in}(v)| \geq \triangle + 1$, for each node $v \in \widehat{T^{in}}$. Claim 5 below implies that for each root S of a tree in the forest F, there are at least $D(S) + f_J(S)$ edges in the graph with heads in

S. Since for each $S' \in \mathscr{L}$, $f_J(S') \geq 1$, we have

$$\begin{aligned}
|E| &\geq \sum_{S \in \ roots(F)} (D(S) + f_J(S)) + (\triangle + 1) \cdot |\widehat{T^{in}}| \\
&\geq \sum_{S \in \ roots(F)} (D(S) + 1) + (\triangle + 1) \cdot |\widehat{T^{in}}| \\
&\geq 1 + \sum_{S \in \ roots(F)} D(S) + \triangle \cdot |\widehat{T^{in}}| + |\widehat{T^{in}}| \\
&= 1 + |\mathscr{L}| + \left|\widetilde{T^{in}}\right| + \triangle \cdot |\widehat{T^{in}}| + |\widehat{T^{in}}| \\
&\geq 1 + |\mathscr{L}| + \left|T^{in}\right|.
\end{aligned}$$

This contradicts the Equation (4.13), implying that $P(f, b^{in}; \alpha, J, B'')$ is $(\alpha, \triangle)$-sparse.

It remains to prove the following claim which gives a lower bound on the number of edges with heads in a given $S \in \mathscr{L}$. For convenience, we drop the subscript J, for example, $f(S)$ below stands for $f_J(S)$. Let $E(X,Y)$ be the number of edges with the tail in set X and the head in set Y.

Claim 5. *If $\triangle$ is a positive integer and $\alpha = 1 + 1/\triangle$, then $E(V,S) \geq D(S) + f(S)$, for any $S \in \mathscr{L}$.*

Proof. We proof the claim by induction on the number of $\mathscr{L}$-descendants of S in F.

Base case: S has no $\mathscr{L}$-descendants in F.
Since for each edge $e \in E$, $0 < x(e) < 1/\alpha$, we have

$$f(S) \ < \ \frac{1}{\alpha} \cdot \left|\delta_E^{in}(S)\right| \ = \ \frac{\triangle}{(\triangle + 1)} \cdot \left|\delta_E^{in}(S)\right| \tag{4.14}$$

For each $v \in T^{in}$, $|\delta_E^{in}(v)| \geq \triangle + 1$ (the first condition of the negation assumption). We split the nodes in $S \cap T^{in}$ into two groups. Let T' be the set of nodes $v \in S \cap T^{in}$ such that $|\delta_E^{in}(v) \cap \delta_E^{in}(S)| \geq \triangle + 1$ and let $T'' = T^{in} \backslash T'$ be the set of nodes $v \in S \cap T^{in}$ such that $|\delta_E^{in}(v) \cap \delta_E^{in}(S)| \leq \triangle$. Hence, $T' \cup T'' = S \cap T^{in}$. Note also that $D(S) = 1 + \left|S \cap T^{in}\right|$.

Since for each $v \in T^{in}$, $|\delta_E^{in}(v)| \geq \triangle + 1$, node $v \in T''$ must have at least one incoming edge which is fully in S. Hence,

$$E(V,S) \geq \left|\delta_E^{in}(S)\right| + \left|T''\right|.$$

For each node $v \in T'$, $|\delta_E^{in}(v) \cap \delta_E^{in}(S)| \geq \triangle + 1$, so $\left|\delta_E^{in}(S)\right| \geq (\triangle + 1)\,|T'|$. Therefore, $1/(\triangle + 1) \cdot \left|\delta_E^{in}(S)\right| \geq |T'|$. Thus, using (4.14),

$$\begin{aligned} E(V,S) &\geq \left(1 - \frac{1}{\triangle + 1}\right) \cdot \left|\delta_E^{in}(S)\right| + \left|T'\right| + |T''| \\ &= \frac{\triangle}{(\triangle + 1)} \cdot \left|\delta_E^{in}(S)\right| + \left|T'\right| + |T''| \\ &> f(S) + \left|S \cap T^{in}\right|. \end{aligned}$$

The above strict inequality implies that

$$E(V,S) \geq f(S) + \left|S \cap T^{in}\right| + 1 = f(S) + D(S).$$

Thus, the claim is true for the base case.

Inductive steps. Take now S in the forest F which has at least one child R in $\mathscr{L}$.

Let $\mathscr{R}$ be the set of all $\mathscr{L}$-children of S and let $R' = \bigcup\{R \in \mathscr{R}\}$ be the union of all children of S in $\mathscr{L}$. Recall the notation that C_S is the union of the set of edges $\delta_E^{in}(R)$ for all $R \in \mathscr{R}$. By inductive hypothesis, we assume that for each child $R \in \mathscr{L}$ of S, we have $E(V,R) \geq D(R) + f(R)$.

Similarly to the base case, let T' be the set of nodes $v \in \left((S \backslash R') \cap T^{in}\right)$ such that $\left|\delta_E^{in}(v) \cap \delta_E^{in}(S)\right| \geq \triangle + 1$ and let T'' be the set of nodes $v \in \left((S \backslash R') \cap T^{in}\right)$ such that $|\delta_E^{in}(v) \cap \delta_E^{in}(S)| \leq \triangle$. Hence, $T' \cup T'' = (S \backslash R') \cap T^{in}$, $T' \cap T'' = \emptyset$, and

$$D(S) = 1 + \left|\left(S \backslash R'\right) \cap T^{in}\right| + \sum_{R \in \mathscr{R}} D(R). \tag{4.15}$$

For each node $v \in T^{in}$, $\left|\delta_E^{in}(v)\right| \geq \triangle + 1$. Therefore, each node $v \in T''$ must have at least 1 edge in $\delta_E^{in}(v)$ from the inside of the set S. Therefore,

$$\begin{aligned} E(V,S) &\geq \sum_{R\in\mathscr{R}} E(V,R) + \left|\delta_E^{in}(S)\backslash C_S\right| + \left|T''\right| \\ &\geq \sum_{R\in\mathscr{R}} (D(R) + f(R)) + \left|\delta_E^{in}(S)\backslash C_S\right| + \left|T''\right| \\ &= \sum_{R\in\mathscr{R}} D(R) + \sum_{R\in\mathscr{R}} f(R) + \left|\delta_E^{in}(S)\backslash C_S\right| + \left|T''\right|. \end{aligned} \tag{4.16}$$

By the definition, for each node $v \in T'$, $\left|\delta_E^{in}(v) \cap \delta_E^{in}(S)\right| \geq \triangle + 1$. Therefore, $\left|\delta_E^{in}(S)\backslash C_S\right| \geq (\triangle + 1)\cdot|T'|$, so $1/(\triangle+1)\cdot\left|\delta_E^{in}(S)\backslash C_S\right| \geq |T'|$. Continuing (4.16) and using (4.15) , we have

$$\begin{aligned} E(V,S) &\geq \sum_{R\in\mathscr{R}} D(R) + \sum_{R\in\mathscr{R}} f(R) + \left(1 - \frac{1}{\triangle+1}\right)\cdot\left|\delta_E^{in}(S)\backslash C_S\right| + \left|T'\right| + \left|T''\right| \\ &= \sum_{R\in\mathscr{R}} D(R) + \sum_{R\in\mathscr{R}} f(R) + \frac{\triangle}{\triangle+1}\cdot\left|\delta_E^{in}(S)\backslash C_S\right| + \left|\left(S\backslash R'\right)\cap T^{in}\right| \\ &= D(S) + \sum_{R\in\mathscr{R}} f(R) + \frac{\triangle}{\triangle+1}\cdot\left|\delta_E^{in}(S)\backslash C_S\right| - 1. \end{aligned} \tag{4.17}$$

Since the constraints corresponding to the sets in $\mathscr{L}$ are tight for x, we have

$$f(S) = x(\delta_E^{in}(S)) = x\left(\delta_E^{in}(S)\cap C_S\right) + x\left(\delta_E^{in}(S)\backslash C_S\right),$$

and

$$\sum_{R\in\mathscr{R}} f(R) = \sum_{R\in\mathscr{R}} x\left(\delta_E^{in}(R)\right) = x(C_S) = x\left(\delta_E^{in}(S)\cap C_S\right) + x\left(C_S\backslash\delta_E^{in}(S)\right).$$

Hence,

$$\begin{aligned} f(S) &= x\left(\delta_E^{in}(S)\cap C_S\right) + x\left(\delta_E^{in}(S)\backslash C_S\right) \\ &\leq x\left(\delta_E^{in}(S)\cap C_S\right) + x\left(\delta_E^{in}(S)\backslash C_S\right) + x\left(C_S\backslash \delta_E^{in}(S)\right) \\ &= \sum_{R\in\mathscr{R}} f(R) + x\left(\delta_E^{in}(S)\backslash C_S\right) \\ &< \sum_{R\in\mathscr{R}} f(R) + \frac{1}{\alpha}\cdot\left|\delta_E^{in}(S)\backslash C_S\right| \\ &= \sum_{R\in\mathscr{R}} f(R) + \frac{\triangle}{\triangle+1}\cdot\left|\delta_E^{in}(S)\backslash C_S\right|. \end{aligned} \tag{4.18}$$

where the last inequality holds since $x(e) < 1/\alpha$ for every $e \in E$.

Thus, (4.17) and (4.18) imply that

$$E(V,S) > D(S) + f(S) - 1.$$

Since $E(V,S)$ is a positive integer, the above strict inequality implies the claimed inequality:

$$E(V,S) \geq D(S) + f(S).$$

□

4.3 DWDCN with both weighted out- and in-degree constraints

In this section, we prove Theorem 3.4, which gives approximation bounds for the DWDCN problem with both out- and in-degree constraints. Let polytope $P(f,b,b^{in}) = P(f,b)\cap P(f,b^{in})$,

that is, polytope $P(f,b,b^{in})$ is defined by the following constraints:

$$\begin{aligned}
x\left(\delta_E^{in}(S)\right) &\geq f(S), \quad \text{for all} \quad \emptyset \neq S \subsetneq V,\\
\sum_{e\in\delta_E^{out}(v)} x(e)w(e) &\leq b(v), \quad \text{for all} \quad v \in B,\\
\sum_{e\in\delta_E^{in}(v)} x(e)w(e) &\leq b^{in}(v), \quad \text{for all} \quad v \in B^{in},\\
0 \leq x(e) &\leq 1, \quad \text{for all} \quad e \in E.
\end{aligned}$$

Given $\alpha \geq 1$, an edge set $J \subseteq E$, and node sets $B' \subseteq B$ and $B'' \subseteq B^{in}$ let the (residual) polytope $P(f,b,b^{in};\alpha,J,B',B'')$ be defined as:

$$\begin{aligned}
x\left(\delta_E^{in}(S)\right) &\geq f_J(S) \equiv f(S) - |\delta_J^{in}(S)|, && \text{for all} \quad \emptyset \neq S \subsetneq V,\\
\sum_{e\in\delta_E^{out}(v)} x(e)w(e) &\leq b_{\alpha,J}(v) \equiv b(v) - w(\delta_J^{out}(v))/\alpha, && \text{for all} \quad v \in B' \subseteq B,\\
\sum_{e\in\delta_E^{in}(v)} x(e)w(e) &\leq b^{in}_{\alpha,J}(v) \equiv b^{in}(v) - w(\delta_J^{in}(v))/\alpha, && \text{for all} \quad v \in B'' \subseteq B^{in},\\
0 \leq x(e) &\leq 1, && \text{for all} \quad e \in E.
\end{aligned}$$

Note that $P(f,b,b^{in};\alpha,\emptyset,B,B^{in})$ is the polytope $P(f,b,b^{in})$.

The algorithm given in Figure 3.1 can be also easily modified to solve the DWDCN problem with both out- and in-degree constraints by considering polytope $P(f,b,b^{in})$ in the beginning of the algorithm, and by finding an optimal basic feasible solution to the LP problem of minimising $\sum_{e\in E} c(e)\cdot x(e)$ over the polytope $P(f,b,b^{in};\alpha,J,B',B'')$ at each iteration of the while loop.

We now use the following extended definition of the sparseness property of a polytope.

Definition 5. *Polytope $P \equiv P(f,b,b^{in};\alpha,J,B',B'')$ is $(\alpha,\triangle,\triangle^{in})$-sparse, if and only if, there exists a node $v \in B'$ with $|\delta_E^{out}(v)| \leq \triangle$, or there exists a node $v \in B''$ with $|\delta_E^{in}(v)| \leq \triangle^{in}$, or for any basic solution $x \in P$, there exists $e \in E$ such that $x(e) = 0$ or $x(e) \geq 1/\alpha$.*

We will prove the following sparseness properties of polytopes $P(f,b,b^{in};\alpha,J,B',B'')$.

Lemma 4.8. *For any $J \subseteq E, B' \subseteq B$, and $B'' \subseteq B^{in}$, polytope $P(f,b,b^{in};\alpha,J,B',B'')$ is $(2,4,4)$-sparse.*

Lemma 4.9. *For any $J \subseteq E, B' \subseteq B$, and $B'' \subseteq B^{in}$, polytope $P(f,b,b^{in};\alpha,J,B',B'')$ is $(3,2,5)$-sparse.*

Lemma 4.10. *If all edges have unit weight, then for any $J \subseteq E, B' \subseteq B$, and $B'' \subseteq B^{in}$, polytope $P(f,b,b^{in};\alpha,J,B',B'')$ is $(2,4,3)$-sparse.*

The following lemma is an extension of Lemma 3.6 to the more general case when the DWDCN problem has both (weighted) out- and in-degree constraints.

Lemma 4.11. *[52] (extension of Lemma 3.6) If for any $J \subseteq E, B' \subseteq B$, and $B'' \subseteq B^{in}$, the polytope $P(f,b,b^{in};\alpha,J,B',B'')$ is $(\alpha,\triangle,\triangle^{in})$-sparse (if non-empty), then* DWDCN *problem with weighted out- and in-degree constraints admits a polynomial-time $(\alpha,(\alpha+\triangle)\cdot b(v),\min\{(\alpha+\triangle^{in}),f_{\max}\}\cdot b^{in}(v))$-approximation algorithm, where $f_{\max}=\max_{S\subseteq V} f(S)$ is the maximum f-value. For the case of unit weights, the* DWDCN *problem admits a polynomial-time $(\alpha,\alpha b(v)+\triangle-1,\min\{\alpha b^{in}(v)+\triangle^{in}-1,f_{\max}\})$-approximation algorithm.*

Theorem 3.4 follows immediately from Lemmas 4.8 – 4.10 and Lemma 4.11. Lemma 4.8 and Lemma 4.11 imply that the DWDCN problem with both weighted out- and in-degree constraints admits a polynomial-time $(2,\ 6b(v),\ \min\{6,f_{\max}\}\cdot b^{in}(v))$-approximation algorithm, where $f_{\max}=\max_{S\subseteq V} f(S)$. Lemma 4.9 and Lemma 4.11 imply that the problem also admits a $(3,\ 5b(v),\ \min\{8,f_{\max}\}\cdot b^{in}(v))$-approximation algorithm. Combining these results together, we prove the first statement of Theorem 3.4. Lemma 4.10 and Lemma 4.11 imply the second statement of Theorem 3.4.

Now it remains to prove Lemmas 4.8, 4.9 and 4.10 to complete the proof of Theorem 3.4. These proofs are given in Section 4.3.1 – 4.3.3. Again we prove these lemmas by contradiction so for convenience we state here the Negation Assumption for the condition of $(\alpha,\triangle,\triangle^{in})$-sparseness. A polytope $P\equiv P(f,b,b^{in};\alpha,J,B',B'')$ is not $(\alpha,\triangle,\triangle^{in})$-sparse, if and only if, the following three conditions are satisfied.

1. $|\delta_E^{out}(v)| \geq \triangle + 1$, for every $v \in B'$.
2. $|\delta_E^{in}(v)| \geq \triangle^{in} + 1$, for every $v \in B''$.
3. There exists a basic solution $x \in P$ such that $0 < x(e) < 1/\alpha$, for every $e \in E$.

Note that the third condition implies that $|\delta_E^{in}(S)| \geq \alpha + 1$ for each $S \in \mathscr{L}$, since $1/\alpha \cdot |\delta_E^{in}(S)| > \sum_{e \in \delta_E^{in}(S)} x(e) = x(\delta_E^{in}(S)) = f_J(S) \geq 1$ (the cut constraints for S is tight and $f_J(S)$ is a positive integer).

Based on standard uncrossing arguments (see [38, 27]), we know that a basic solution of $P(f, b, b^{in}; \alpha, J, B', B'')$ is characterised by a laminar family of tight constraints. The following lemma is given in [52], which is an extension of Lemma 4.4.

Lemma 4.12. *[52] For any basic solution $x \in P(f, b, b^{in}; \alpha, J, B', B'')$ with $0 < x(e) < 1$ for all $e \in E$, there exists a laminar family $\mathscr{L}$ on V, $T \subseteq B'$ and $T^{in} \subseteq B''$, such that $f_J(S) \geq 1$ for all $S \in \mathscr{L}$, and such that x is the unique solution to the system of linear equations:*

$$\begin{aligned} x(\delta_E^{in}(S)) &= f_J(S), && \text{for all } S \in \mathscr{L}, \\ \sum_{e \in \delta_E^{out}(v)} x(e)w(e) &= b_{\alpha,J}(v), && \text{for all } v \in T, \\ \sum_{e \in \delta_E^{in}(v)} x(e)w(e) &= b^{in}_{\alpha,J}(v), && \text{for all } v \in T^{in}, \end{aligned}$$

and $|\mathscr{L}| + |T| + |T^{in}| = |E|$. In particular, the characteristic vectors of $\{\delta_E^{in}(S) : S \in \mathscr{L}\}$ are linearly independent.

As in Section 4.1, we define a child-parent relation on the members of $\mathscr{L} \cup T \cup T^{in}$. For $S \in \mathscr{L}$ or $v \in T$ or $v' \in T^{in}$, its parent S' is the inclusion minimal member of $\mathscr{L}$ properly containing it, i.e, the set S' is the smallest set in $\mathscr{L}$ containing S or v or v'. Note that when $v \in T \cup T^{in}$ and $\{v\} \in \mathscr{L}$, then v is a child of $\{v\}$. This relation defines a forest F with the "vertex" set $\mathscr{L} \cup T \cup T^{in}$.

In the following subsections we will prove Lemmas 4.8 – 4.10.

4.3.1 $(2,4,4)$-sparseness of $P(f,b,b^{in};\alpha,J,B',B'')$ - proof of Lemma 4.8

We prove that $P \equiv P(f,b,b^{in};\alpha,J,B',B'')$ is $(2,4,4)$-sparse by contradiction, assuming the Negation Assumption for $(2,4,4)$-sparseness and taking a basic solution $x \in P$ satisfying Condition 3 of this assumption. For this basic solution, let $\mathscr{L}$, T and T^{in} be as in Lemma 4.12. Hence, we have $|\delta_E^{in}(S)| \geq 3$ for all $S \in \mathscr{L}$ (from Condition 3 of the Negation Assumption), $|\delta_E^{out}(v)| \geq 5$ for all $v \in T$ (from Condition 1 of the Negation Assumption) and $|\delta_E^{in}(v)| \geq 5$ for all $v \in T^{in}$ (from Condition 2 of the Negation Assumption).

Based on the initial assignment given in Section 4.1.2, we will show in Claim 6 below that a $(2,3)$-token reassignment scheme is feasible (see Definition 3), i.e, we can make an assignment such that each root S of a tree in the forest F gets at least 3 S-tokens and each descendant of S gets at least 2 S-tokens.

We obtain a contradiction as follows. Let the root tokens refer to the S-tokens for all $S \in \mathscr{L}$, which are roots of trees in the forest F. Let $\hat{T} \subseteq T$ and $\widehat{T^{in}} \subseteq T^{in}$ be the set of nodes in T and T^{in}, respectively, which correspond to single-vertex trees in F and let $\tilde{T} = T \setminus \hat{T}$ and $\widetilde{T^{in}} = T^{in} \setminus \widehat{T^{in}}$ be the set of nodes in T and T^{in}, receptively, which have a parent in F. Note that the two node-tokens of $v \in \hat{T}$ and 1/2 tail-tokens of each edge (v,u) such that $v \in \hat{T}$ are not S-tokens, for any $S \in \mathscr{L}$. Moreover, a head-token of each edge (u,v) such that $v \in \widehat{T^{in}}$ are not S-tokens, for any $S \in \mathscr{L}$. Therefore, the Negation Assumption implies that the number of the root tokens in F is at most $2|E| - (2+5/2)|\hat{T}| - 5|\widehat{T^{in}}|$. The existence of a $(2,3)$-token reassignment scheme implies that there are at least $2(|\mathscr{L}| + |\tilde{T}| + |\widetilde{T^{in}}|) + 1$ root-tokens. Hence,

$$2|E| - 9/2|\hat{T}| - 5|\widehat{T^{in}}| \geq 2(|\mathscr{L}| + |\tilde{T}| + |\widetilde{T^{in}}|) + 1.$$

Therefore,

$$\begin{aligned} 2|E| &\geq 2(|\mathscr{L}| + |\tilde{T}| + |\widetilde{T^{in}}| + |\hat{T}| + |\widehat{T^{in}}|) + 1 + \frac{5}{2}|\hat{T}| + 3|\widehat{T^{in}}| \\ &= 2(|\mathscr{L}| + |T| + |T^{in}|) + 1 + \frac{5}{2}|\hat{T}| + 3|\widehat{T^{in}}| \\ &> 2(|\mathscr{L}| + |T| + |T^{in}|). \end{aligned}$$

This contradicts $|E| = |\mathscr{L}| + |T| + |T^{in}|$, which implies that polytope $P(f,b,b^{in};\alpha,J,B',B'')$ is $(2,4,4)$-sparse.

The remaining part of the proof of Lemma 4.8 is the proof of the following Claim 6.

Claim 6. *A $(2,3)$-token reassignment scheme is feasible if polytope $P(f,b,b^{in};\alpha,J,B',B'')$ is not $(2,4,4)$-sparse.*

Proof. We first perform the initial assignment as given in Section 4.1.2. We show in Claim 1 (see Section 4.1.2) that a $(2,3)$-token reassignment scheme is feasible, if polytope $P(f,b;\alpha,J,B')$ is not $(2,4)$-sparse. Therefore, the proof of Claim 1 and the Negation Assumption for $(2,4,4)$-sparseness imply that a $(2,3)$-token reassignment scheme is also feasible, if $T^{in} = \emptyset$. Hence to prove this claim (Claim 6), it is enough to show that for each $S \in \mathscr{L}$, we have a sufficient number of S-tokens to assign $v \in T^{in}$, if v is a child of S. Therefore, in the rest of this proof, we will focus only on nodes $v \in T^{in}$ and show how these nodes get 2 S-tokens.

First consider the base case of Claim 1 and assume that there is at least one node $v \in T^{in}$ in S, i.e, $|S \cap T^{in}| \geq 1$. Recall that in the base case, we assigned to S, the head-tokens from any three edges in $\delta_E^{in}(S)$. Let p be the set of these edges. Hence, $|p| = 3$. The Negation Assumption for $(2,4,4)$-sparseness implies that for each node $v \in T^{in}$, we have $|\delta_E^{in}(v)| \geq 5$. Therefore, $|\delta_E^{in}(v) \setminus p| \geq 2$. Observe that the head-tokens of edges in $\delta_E^{in}(v) \setminus p$ have not been used. We assign to v the head-tokens from any two edges in $\delta_E^{in}(v) \setminus p$. All these head-tokens (2 tokens) are S-tokens by Definition 2, so each node $v \in S \cap T^{in}$ gets the required number of S-tokens.

Now consider the inductive step of Claim 1 and assume that $|(S \setminus R') \cap T^{in}| \geq 1$, where R' is the union of all children of S in $\mathscr{L}$. Observe that during the inductive steps in Claim 1, we did not use any head-token of edge (u,v) such that $u,v \in S$. Only the head-tokens from edges in $\delta_E^{in}(S) \setminus C_S$ were used in order to make an assignment for set S. Also note that among these edges in $\delta_E^{in}(S) \setminus C_S$, only (at most) two edges were chosen and their head-tokens were assigned to S. Let p be the set of these edges. So $|p| \leq 2$. This implies that for each node

$v \in (S \backslash R') \cap T^{in}$, we have $|\delta_E^{in}(v) \backslash p| \geq 3$. Again, the head-tokens of edges in $\delta_E^{in}(v) \backslash p$ have never been used. Hence, we have a sufficient number of the S-tokens. We assign to v the head-tokens from any two edges $e \in \delta_E^{in}(S) \backslash p$. All these head-tokens (2 tokens) are S-tokens by Definition 2, so each node $v \in S \cap T^{in}$ gets the required number of S-tokens.

Thus, a $(2,3)$-token reassignment scheme is feasible, if polytope $P(f,b,b^{in};\alpha,J,B',B'')$ is not $(2,4,4)$-sparse.

□

4.3.2 $(3,2,5)$-sparseness of $P(f,b,b^{in};\alpha,J,B',B'')$ - proof of Lemma 4.9

We now prove that polytope $P \equiv P(f,b,b^{in};\alpha,J,B',B'')$ is $(3,2,5)$-sparse, assuming Negation Assumption for $(3,2,5)$-sparseness and taking a basic solution x satisfying Condition 3 of this assumption. For this basic solution $x \in P$, let $\mathscr{L}$ be a laminar family, and sets T and T^{in} be defined as in Lemma 4.12. Hence, we have $|\delta_E^{in}(S)| \geq 4$ for all $S \in \mathscr{L}$ (from Condition 3 of the Negation Assumption), $|\delta_E^{out}(v)| \geq 3$ for all $v \in T$ (from Condition 1 of the Negation Assumption) and $|\delta_E^{in}(v)| \geq 6$ for all $v \in T^{in}$ (from Condition 2 of the Negation Assumption).

Based on the initial assignment given in Section 4.1.3, we will show in Claim 7 below that a $(2,4)$-token reassignment scheme is feasible, i.e, we can make an assignment such that each root S of a tree in the forest F gets at least 4 S-tokens and each descendant of S gets at least 2 S-tokens.

We obtain a contradiction as before. Let the root tokens refer to the S-tokens for all $S \in \mathscr{L}$, which are roots of trees in the forest F. Let $\hat{T} \subseteq T$ and $\widehat{T^{in}} \subseteq T^{in}$ be the set of nodes in T and T^{in}, respectively, which correspond to single-vertex trees in F and let $\tilde{T} = T \backslash \hat{T}$ and $\widetilde{T^{in}} = T^{in} \backslash \widehat{T^{in}}$ be the set of nodes in T and T^{in}, receptively, which have a parent in F. The number of the root tokens in F is at most $2|E| - 3|\hat{T}| - 6|\widehat{T^{in}}|$ since the three node-tokens of $v \in \hat{T}$ are not S-tokens for any $S \in \mathscr{L}$ and a head-token of each edge (u,v) such that $v \in \widehat{T^{in}}$

are not S-tokens for any $S \in \mathscr{L}$. The feasibility of a $(2,4)$-token reassignment scheme implies that there are at least $2(|\mathscr{L}| + |\tilde{T}| + |\widetilde{T^{in}}|) + 2$ root-tokens. Hence,

$$2|E| - 3|\hat{T}| - 6|\widehat{T^{in}}| \geq 2(|\mathscr{L}| + |\tilde{T}| + |\widetilde{T^{in}}|) + 2.$$

Therefore,

$$\begin{aligned} 2|E| &\geq 2(|\mathscr{L}| + |\tilde{T}| + |\widetilde{T^{in}}| + |\hat{T}| + |\widehat{T^{in}}|) + 2 + |\hat{T}| + 4|\widehat{T^{in}}| \\ &= 2(|\mathscr{L}| + |T| + |T^{in}|) + 2 + |\hat{T}| + 4|\widehat{T^{in}}| \\ &> 2(|\mathscr{L}| + |T| + |T^{in}|). \end{aligned}$$

This contradicts $|E| = |\mathscr{L}| + |T| + |T^{in}|$, which implies that polytope $P(f, b, b^{in}; \alpha, J, B', B'')$ is $(3,2,5)$-sparse.

It remains to prove Claim 7 to complete the proof of Lemma 4.9.

Claim 7. *A $(2,4)$-token reassignment scheme is feasible if polytope $P(f, b, b^{in}; \alpha, J, B', B'')$ is not $(3,2,5)$-sparse.*

Proof. We first perform the initial assignment as given in Section 4.1.3. Analogously to the proof of Claim 6, we can show that a $(2,4)$-token reassignment scheme is feasible. For completeness, we will give the proof here.

First consider the base case of Claim 2 and assume that there is at least one node $v \in T^{in}$ in S, i.e, $|S \cap T^{in}| \geq 1$. Recall that we assigned to S, the head-tokens from any four edges in $\delta_E^{in}(S)$. Let p be the set of these edges. Hence, $|p| = 4$. The Negation Assumption for $(3,2,5)$-sparseness implies that for each node $v \in T^{in}$, we have $|\delta_E^{in}(v)| \geq 6$. Therefore, $|\delta_E^{in}(v) \setminus p| \geq 2$. We assign to v the head-tokens from any two edges in $\delta_E^{in}(v) \setminus p$. All these head-tokens (2 tokens) are S-tokens by Definition 2, so each node $v \in S \cap T^{in}$ gets the required number of S-tokens.

Now consider the inductive step of Claim 2 and assume that $|(S\backslash R')\cap T^{in}|\geq 1$, where R' is the union of all children of S in $\mathscr{L}$. Observe that during the inductive steps in Claim 2, we did not use any head-token of edge (u,v) such that $u,v\in S$. Only the head-tokens from $e\in\delta_E^{in}(S)\backslash C_S$ were used during the inductive steps in order to make an assignment for S. Also note that among these edges $e\in\delta_E^{in}(S)\backslash C_S$, only (at most) two edges were chosen and their head-tokens were assigned to S. Let p be the set of these edges. So $|p|\leq 2$. This implies that for each node $v\in(S\backslash R')\cap T^{in}$, we have $|\delta_E^{in}(v)\backslash p|\geq 4$. Hence, we have a sufficient number of S-tokens. We assign to v the head-tokens from any two edges $e\in\delta_E^{in}(v)\backslash p$. All these head-tokens (2 tokens) are S-tokens by Definition 2, so each node $v\in S\cap T^{in}$ gets the required number of the S-tokens.

Thus, a $(2,4)$-token reassignment scheme is feasible, if polytope $P(f,b,b^{in};\alpha,J,B',B'')$ is not $(3,2,4)$-sparse. □

4.3.3 $(2,4,3)$-sparseness of $P(f,b,b^{in};\alpha,J,B',B'')$ - proof of Lemma 4.10

We prove that polytope $P\equiv P(f,b,b^{in};\alpha,J,B',B'')$ is $(2,4,3)$-sparse assuming Negation Assumption for $(2,4,3)$-sparseness and taking a basic solution x satisfying Condition 3 of this assumption. For this basic solution $x\in P$, let $\mathscr{L}$ be a laminar family, and sets T and T^{in} be defined as in Lemma 4.12. Hence, we have $|\delta_E^{in}(S)|\geq 3$ for all $S\in\mathscr{L}$, $|\delta_E^{out}(v)|\geq 5$ for all $v\in T$ and $|\delta_E^{in}(v)|\geq 4$ for all $v\in T^{in}$.

Here we perform the following initial assignment of $2|E|$ tokens and use the same definition of S-tokens as given in Definition 2.

Initial assignment

For each $e=(v,u)\in E$ such that $v\notin T$, we designate 1 tokens as the head-tokens of edge e and 1 tokens as the tail-tokens of edge e. For each $v\in T$, there are $|\delta_E^{out}(v)|\geq 5$ outgoing edges from node v. We designate 2 tokens as the node-tokens of v and 1 tokens as the head-tokens of each edge $e\in\delta_E^{out}(v)$. We also designate the tail-tokens for the edges in $\delta_E^{out}(v)$ according to the following three cases.

1. **If v has exactly one $\mathscr{L}$-sibling R in forest F and there is at least one edge (v,x) such that $x \in R$:** we choose one of these edges and designate 1 tokens as the tail-tokens of this edge. We then designate $1/2$ tokens as the tail-tokens for each of the other edges in $\delta_E^{out}(v)$.

2. **If v has no $\mathscr{L}$-sibling in forest F and there is at least one edge (v,x) such that $x \in T^{in}$:** we choose one of these edges and designate 1 tokens as the tail-tokens of this edge. We then designate $1/2$ tokens as the tail-tokens for each of the other edges in $\delta_E^{out}(v)$.

3. **Otherwise:** For each (v,u), we designate $1/2$ tokens as the tail-tokens.

It can be verified that at most $2|E|$ tokens are used in this initial assignment. Note that node-tokens at $v \in T$ are fixed at v and never used during the reassignment process.

Using this initial assignment, we will show in Claim 8 that a $(2,3)$-token reassignment is feasible (see Definition 3). We obtain a contradiction as follows. Let the root tokens refer to the S-tokens for all $S \in \mathscr{L}$, which are roots of trees in the forest F. Let $\hat{T} \subseteq T$ and $\widehat{T^{in}} \subseteq T^{in}$ be the set of nodes in T and T^{in}, respectively, which correspond to single-vertex trees in F and let $\tilde{T} = T \setminus \hat{T}$ and $\widetilde{T^{in}} = T^{in} \setminus \widehat{T^{in}}$ be the set of nodes in T and T^{in}, respectively, which have a parent in F. Note that the two node-tokens of $v \in \hat{T}$ and 1/2 tail-tokens of each edge (v,u) such that $v \in \hat{T}$ are not S-tokens for any $S \in \mathscr{L}$. Moreover, a head-token of each edge (u,v) such that $v \in \widehat{T^{in}}$ are not S-tokens for any $S \in \mathscr{L}$. Therefore, the Negation Assumption implies that the number of the root tokens in F is at most $2|E| - (2 + 5 \cdot (1/2))|\hat{T}| - 4|\widehat{T^{in}}|$. The feasibility of a $(2,3)$-token reassignment scheme implies that there are at least $2(|\mathscr{L}| + |\tilde{T}| + |\widetilde{T^{in}}|) + 1$ root-tokens. Hence,

$$2|E| - \frac{9}{2}|\hat{T}| - 4|\widehat{T^{in}}| \;\geq\; 2(|\mathscr{L}| + |\tilde{T}| + |\widetilde{T^{in}}|) + 1.$$

Therefore,

$$\begin{aligned} 2|E| &\geq 2(|\mathscr{L}|+|\tilde{T}|+|\widetilde{T^{in}}|+|\hat{T}|+|\widehat{T^{in}}|)+1+\frac{5}{2}|\hat{T}|+2|\widehat{T^{in}}| \\ &= 2(|\mathscr{L}|+|T|+|T^{in}|)+1+\frac{5}{2}|\hat{T}|+2|\widehat{T^{in}}| \\ &> 2(|\mathscr{L}|+|T|+|T^{in}|). \end{aligned}$$

This contradicts $|E| = |\mathscr{L}|+|T|+|T^{in}|$, which implies that polytope $P(f,b,b^{in};\alpha,J,B',B'')$ is $(2,4,3)$-sparse.

The remaining part of the proof of Lemma 4.10 is the proof of the following claim.

Claim 8. *A $(2,3)$-token reassignment scheme is feasible, if polytope $P(f,b,b^{in};\alpha,J,B',B'')$ is not $(2,4,3)$-sparse and the polytope is constructed for unit-weight graph $G=(V,E,w,b)$, i.e, $w(e)=1$, for all edges $e \in E$.*

Proof. Based on the initial assignment above, we show that a $(2,3)$-token reassignment scheme is feasible. The proof is given by induction on the number of $\mathscr{L}$-descendants of in F. Recall that each node $v \in T$ has two node-tokens, which are fixed at node v and never used during the reassignment process. By Definition 2, all these node-tokens are S-tokens for each $S \in \mathscr{L}$ such that $v \in S$. Therefore $v \in T$ always gets the required number of S-tokens if v is a descendant of S in F.

Base case: S has no $\mathscr{L}$-descendants in F.

We consider separately the cases when $S \cap T^{in} = \emptyset$, $|S \cap T^{in}| \geq 2$, and $|S \cap T^{in}| = 1$. Consider first case that $S \cap T^{in} = \emptyset$. We assign to S the head-tokens from any three edges $(u,v) \in \delta_E^{in}(S)$. This way S gets 3 head-tokens (3 tokens). All these head-tokens are S-tokens by Definition 2. Hence, S gets the required number of S-tokens.

Now consider the case when $|S \cap T^{in}| \geq 2$. By the Negation Assumption, we have $|\delta_E^{in}(v)| \geq 4$, for each node $v \in T^{in}$. This implies that we have at least $4 \cdot |S \cap T^{in}|$ head-tokens in S. By

Definition 2, all these head-tokens ($4 \cdot |S \cap T^{in}|$ tokens) are S-tokens. The number of S-tokens we need for the assignment is at least $2 \cdot |S \cap T^{in}| + 3$, that is, 2 S-tokens for each $v \in S \cap T^{in}$ and 3 S-tokens for set S. Since $4 \cdot |S \cap T^{in}| > 2|S \cap T^{in}| + 3$, if $|S \cap T^{in}| \geq 2$, we have a sufficient number of S-tokens in S for the assignment.

Now consider the case when $|S \cap T^{in}| = 1$. Let $v \in S \cap T^{in}$. Now we consider the following sub-cases, when $|\delta_E^{in}(S) \cap \delta_E^{in}(v)| \leq 2$ and $|\delta_E^{in}(S) \cap \delta_E^{in}(v)| \geq 3$. First consider the sub-case that $|\delta_E^{in}(S) \cap \delta_E^{in}(v)| \leq 2$. The Negation Assumption implies that $|\delta_E^{in}(S)| \geq 3$ and also that $|\delta_E^{in}(v) \setminus \delta_E^{in}(S)| \geq 2$ because $|\delta_E^{in}(S) \cap \delta_E^{in}(v)| \leq 2$ and $|\delta_E^{in}(v)| \geq 4$ for each node $v \in T^{in}$. Therefore, we have at least 5 head-tokens in set S. By Definition 2, all these head-tokens (5 tokens) are S-tokens. Hence, using these S-tokens, we can assign set S, 3 S-tokens and assign v, 2 S-tokens. Thus, we get an assignment as required.

Now consider the other case that $|\delta_E^{in}(S) \cap \delta_E^{in}(v)| \geq 3$. Note that $\delta_E^{in}(S) \neq \delta_E^{in}(v)$ as equations of S and v are linearly independent. This implies that if $|\delta_E^{in}(S)| = |\delta_E^{in}(v)|$, there should be at least one edge in $\delta_E^{in}(S) \setminus \delta_E^{in}(v)$, i.e, $|\delta_E^{in}(S) \setminus \delta_E^{in}(v)| \geq 1$ or there should be at least one edge in $\delta_E^{in}(v) \setminus \delta_E^{in}(S)$, i.e, $|\delta_E^{in}(v) \setminus \delta_E^{in}(S)| \geq 1$. Consider first that $|\delta_E^{in}(S) \setminus \delta_E^{in}(v)| \geq 1$. Note that for each node $v \in T^{in}$, $|\delta_E^{in}(v)| \geq 4$. Therefore, we have at least 4 head-tokens from $e \in \delta_E^{in}(v)$ and at least 1 head-token from $e \in \delta_E^{in}(S) \setminus \delta_E^{in}(v)$. Hence, in total, we have at least 5 head-tokens in S. By Definition 2, all these head-tokens (5 tokens) are S-tokens. Hence, using these S-tokens, we can assign set S, 3 S-tokens and assign v, 2 S-tokens. Thus, we get an assignment as required. Now consider the other case that $|\delta_E^{in}(v) \setminus \delta_E^{in}(S)| \geq 1$. Observe that if $(u,v) \in \delta_E^{in}(v) \setminus \delta_E^{in}(S)$, then, $u \in S$ and $v \in S$. Since $|\delta_E^{in}(S)| \geq 3$, we have at least 3 head-tokens and since $|\delta_E^{in}(v) \setminus \delta_E^{in}(S)| \geq 1$, we have at least 1 head-token and 1 tail-token. Recall that during the initial assignment if there exists $(u,v) \in \delta_E^{out}(u)$ such that $u \in T$, $v \in T^{in}$ and u has no $\mathscr{L}$-sibling in forest F, then we assigned 1 token as the tail-token of this edge. Therefore, in either case $u \in T$ or $u \notin T$, edge (u,v) has a tail-token (1 token). Hence, in set S, we have at least 4 head-tokens and at least 1 tail-token. By Definition 2 all four head-tokens (4 tokens) and the tail-token (1 tokens) are S-tokens. Thus, we have enough number of S-tokens; we can assign S, 3 head-tokens and assign v 1 head-token and 1 tail-token.

Inductive steps. Take now S in the forest F which has at least one child R in $\mathscr{L}$. Recall the notation that C_S is the union of the sets of edges $\delta_E^{in}(R)$ for all children R of S in $\mathscr{L}$. By inductive hypothesis, we assume that for each child $R \in \mathscr{L}$ of S, R has at least 3 R-tokens and each descendant of R has at least 2 R-tokens. We consider separately the case when S has exactly one child R in $\mathscr{L}$ and the case when S has at least two children in $\mathscr{L}$.

S has exactly 1 child R in $\mathscr{L}$: We move 1 tokens from R to S. Now R has 2 R-tokens and S has 1 R-tokens. Each descendant of R also has 2 R-tokens. By Definition 2, all these R-tokens are S-tokens. Thus R and its descendants have the required number of S-tokens, but S needs 2 additional S-tokens. We consider separately the case when $|(S\backslash R) \cap T^{in}| \geq 1$ and when $|(S\backslash R) \cap T^{in}| = 0$, i.e, $(S\backslash R) \cap T^{in} = \emptyset$.

First consider the case when $|(S\backslash R) \cap T^{in}| \geq 1$. The number of S-tokens we need for the assignment is at least $2 \cdot |(S\backslash R) \cap T^{in}| + 2$, that is, 2 S-tokens for each $v \in (S\backslash R) \cap T^{in}$ and 2 S-tokens for set S. Recall that for each node $v \in T^{in}$, we have $|\delta_E^{in}(v)| \geq 4$ by the Negation Assumption. This implies that we have at least $4 \cdot |(S\backslash R) \cap T^{in}|$ head-tokens in S. By Definition 2, all these head-tokens are S-tokens. Since $4 \cdot |(S\backslash R) \cap T^{in}| \geq 2 + 2|(S\backslash R) \cap T^{in}|$, if $|(S\backslash R) \cap T^{in}| \geq 1$, we have enough S-tokens for the assignment.

Consider now the other case when $|(S\backslash R) \cap T^{in}| = 0$. Lemma 4.5 implies that we have only the following sub-cases: (i) both sets $C_S\backslash\delta_E^{in}(S)$ and $\delta_E^{in}(S)\backslash C_S$ have at least 1 edge, (ii) $C_S\backslash\delta_E^{in}(S)$ is empty and $\delta_E^{in}(S)\backslash C_S$ has at least 3 edges, (iii) $\delta_E^{in}(S)\backslash C_S$ is empty and $C_S\backslash\delta_E^{in}(S)$ has at least 3 edges. Recall that S gets 1 R-token from R. This token is also an S-token, so, S needs 2 more S-tokens.

(i) Both sets $C_S\backslash\delta_E^{in}(S)$ and $\delta_E^{in}(S)\backslash C_S$ have at least 1 edge.

We assign to S a head-token from any edge $e \in \delta_E^{in}(S)\backslash C_S$ and a tail-token from any edge $(u,x) \in C_S\backslash\delta_E^{in}(S)$. Recall that during the initial assignment if there exists an edge $(v,z) \in \delta_E^{out}(v)$ such that $v \in T$ and $z \in R$, then we assigned 1 token as the tail-token of this edge. Therefore, in any case, either $v \in T$ or $v \notin T$, S gets a tail-token (1 token) from $(v,z) \in C_S\backslash\delta_E^{in}(S)$. By Definition 2, the head-token (1 token) and the tail-token (1 token) are S-tokens. Therefore, S gets 2 S-tokens.

(ii) $C_S \setminus \delta_E^{in}(S)$ is empty and $\delta_E^{in}(S) \setminus C_S$ has at least 3 edges.

We assign to S the head-tokens from any three edges $e \in \delta_E^{in}(S) \setminus C_S$. This way S gets 3 head-tokens (3 tokens), which are by definition S-tokens. Thus, S get the required number of S-tokens.

(iii) $\delta_E^{in}(S) \setminus C_S$ is empty and $C_S \setminus \delta_E^{in}(S)$ has at least 3 edges.

We assign to S the tail-tokens from all edges $(v,u) \in C_S \setminus \delta_E^{in}(S)$. The tails of these edges may or may not be in T. If none of them is in T, then S gets 3 tail-tokens (3 tokens). All these tail-tokens are S-tokens. Hence, S gets an assignment as required. Now consider the case that none of edges $(v,z) \in C_S \setminus \delta_E^{in}(S)$ are such that $v \notin T$, i.e, all edges $(v,z) \in C_S \setminus \delta_E^{in}(S)$ are such that $v \in T$. Recall that during the initial assignment if there exists $(v,z) \in \delta_E^{out}(v)$ such that $z \in R$ and $v \in T$, then we assigned 1 tail-token to this edge, and 1/2 tail-token to the remaining edges in $\delta_E^{out}(v)$. Therefore, S gets at least three tail-tokens ($1+1/2+1/2=2$ tokens). These tail-tokens (2 tokens) are S-tokens by the definition. Hence, S gets an assignment as required.

S has at least 2 children in $\mathscr{L}$: Let R' be the union of all children R of $S \in \mathscr{L}$. For each child $R \in \mathscr{L}$ of S, we move 1 token from R to S. Each child R now has 2 R-tokens and S has at least 2 R-tokens. Each descendant of R also has 2 R-tokens. By definition, all these R-tokens are S-tokens. Thus R and its descendants have the required number of S-tokens, but S needs at least 1 additional S-token and at least 2 S-tokens for each node $v \in T^{in}$ in $S \setminus R'$. We consider the following two cases when $|(S \setminus R') \cap T^{in}| \geq 1$ and when $|(S \setminus R') \cap T^{in}| = 0$.

First consider that $|(S \setminus R') \cap T^{in}| \geq 1$. The number of S-tokens we need for the assignment is at least $2 \cdot |(S \setminus R) \cap T^{in}| + 1$, that is, 2 S-tokens for each $v \in (S \setminus R) \cap T^{in}$ and 1 S-tokens for set S. Note that for each node $v \in T^{in}$, $|\delta_E^{in}(v)| \geq 4$. This implies that we have at least $4 \cdot |(S \setminus R') \cap T^{in}|$ head-tokens in S, if $|(S \setminus R') \cap T^{in}| \geq 1$. By Definition 2, all these head-tokens ($4 \cdot |(S \setminus R') \cap T^{in}|$ tokens) are S-tokens. Since $(4 \cdot |(S \setminus R') \cap T^{in}| > 2 \cdot |(S \setminus R') \cap T^{in}| + 1$, if $|(S \setminus R') \cap T^{in}| \geq 1$, we have enough S-tokens to for the assignment.

Now consider the other case when $|(S \setminus R') \cap T^{in}| = 0$. Lemma 4.5 implies that we have only the following sub-cases: (i) both sets $C_S \setminus \delta_E^{in}(S)$ and $\delta_E^{in}(S) \setminus C_S$ have at least 1 edge, (ii) $C_S \setminus \delta_E^{in}(S)$ is empty and $\delta_E^{in}(S) \setminus C_S$ has at least 3 edges, (iii) $\delta_E^{in}(S) \setminus C_S$ is empty and

$C_S \backslash \delta_E^{in}(S)$ has at least 3 edges. Recall that S gets at least 2 S-tokens from its $\mathscr{L}$-children, So, S needs at least 1 S-token.

(i) Both sets $C_S \backslash \delta_E^{in}(S)$ and $\delta_E^{in}(S) \backslash C_S$ have at least 1 edge.
We assign to S the head-token from one edge $e \in \delta_E^{in}(S) \backslash C_S$. This head-tokens (1 token) is an S-token. Hence S gets the required number of S-tokens.

(ii) $C_S \backslash \delta_E^{in}(S)$ is empty and $\delta_E^{in}(S) \backslash C_S$ has at least 3 edges.
We assign to S the head-token from one edge $e \in \delta_E^{in}(S) \backslash C_S$. This head-tokens (1 token) is an S-token. Hence S gets the required number of S-tokens.

(iii) $\delta_E^{in}(S) \backslash C_S$ is empty and $C_S \backslash \delta_E^{in}(S)$ has at least 3 edges.
We assign to S the tail-tokens from all edges $(v,u) \in C_S \backslash \delta_E^{in}(S)$. The tail-tokens can be either from (a) $(v,u) \in C_S \backslash \delta_E^{in}(S)$ such that v is not in any child of S, or (b) $(v,u) \in C_S \backslash \delta_E^{in}(S)$ such that v is in child R of S and u is in another child R' of S. Clearly, the tail-tokens in the case (a) have not been assigned before since they are not R-tokens for any child R of S. The node v in the case (b) has been considered previously as v is inside R. However, in this case, the tail-tokens assigned to edge (v,u) have not been used as by definition, they are not R-tokens for the child R of S (the tail-tokens from (x,y) are R-tokens only if both x,y are in R). By definition, all these tail-tokens are S-tokens. Each tail-token in $(v,u) \in C_S \backslash \delta_E^{in}(S)$ has either 1 token (if $v \notin T$) or $1/2$ token (if $v \in T$). Therefore, S gets at least $3 \cdot (1/2) > 1$ S-tokens. Thus S gets the required number of S-tokens.

□

Chapter 5

Approximation bounds for crossing supermodular connectivity requirements

In this chapter, we consider the DWDCN problems under the crossing supermodular connectivity requirements, and derive new improved approximation bounds for these problems. In Section 5.1, we consider the DWDCN problem with weighted out-degree constraints and the DWDCN problem with weighted in-degree constraints. In Section 5.2, we consider DWDCN problems with both weighted out- and in-degree constraints.

The following lemma, which reduces the case of crossing supermodular function to intersecting supermodular is a known fact; see for example [48, 52]. For completeness, we include a proof of this lemma.

Lemma 5.1. *Let f be a crossing supermodular set function on V and let $r \in V$ be an arbitrary fixed node. Let $f^{out}(S) = f(S)$, if $S \in \{A \subset V : r \notin A\}$, and $f^{out}(S) = 0$ otherwise, and let $f^{in}(S) = f(V \setminus S)$, if $S \in \{A \subset V : r \notin A\}$, and $f^{in}(S) = 0$ otherwise. Then f^{out} and f^{in} are intersecting supermodular set functions on $V \setminus \{r\}$. Furthermore, graph H is f-connected, if and only if, H is f^{out}-connected and the reverse graph H_R of H is f^{in}-connected.*

Proof. We first show that f^{out} and f^{in} are intersecting supermodular set functions on $V\backslash\{r\}$. Let $\sigma = \{A \subset V : r \notin A\}$. Suppose that $X, Y \in \sigma$ such that $X \cap Y \neq \emptyset$. Since $r \notin X$ and $r \notin Y$, $X \cup Y \neq V$. If $X \subseteq Y$ or $Y \subseteq X$, then trivially $f(X) + f(Y) \leq f(X \cap Y) + f(X \cup Y)$. If $X\backslash Y \neq \emptyset$ and $Y\backslash X \neq \emptyset$, then by the definition of f^{out} and crossing supermodularity condition of f, we get

$$f^{out}(X) + f^{out}(Y) = f(X) + f(Y) \leq f(X \cap Y) + f(X \cup Y) = f^{out}(X \cap Y) + f^{out}(X \cup Y).$$

Hence, f^{out} is intersecting supermodular on $V\backslash\{r\}$. Similarly, we can get

$$\begin{aligned} f^{in}(X) + f^{in}(Y) &= f(V\backslash X) + f(V\backslash Y) = f(\overline{X}) + f(\overline{Y}) \leq f(\overline{X} \cup \overline{Y}) + f(\overline{X} \cap \overline{Y}) \\ &= f(V\backslash(X \cap Y)) + f(V\backslash(X \cup Y)) = f^{in}(X \cap Y) + f^{in}(X \cup Y), \end{aligned}$$

where the inequality follows from supermodularity condition of f and the fact that $\overline{X}$ and $\overline{Y}$ are a crossing pair. Thus, f^{in} is intersecting supermodular on $V\backslash\{r\}$.

We now show that H is f^{out}-connected and H_R is f^{in}-connected, if H is f-connected. Assume that H is f-connected. Let S be any non-empty proper subset of V. We show that H is f^{out}-connected, i.e, $|\delta^{in}_H(S)| \geq f^{out}(S)$. If $r \in S$, then $f^{out}(S) = 0$, so $|\delta^{in}_H(S)| \geq f^{out}(S)$. If $r \notin S$, then since H is f-connected,

$$|\delta^{in}_H(S)| \geq f(S) = f^{out}(S).$$

We now show that H_R is f^{in}-connected, i.e, $|\delta^{in}_{H_R}(S)| \geq f^{in}(S)$. If $r \in S$, then $f^{in}(S) = 0 \leq |\delta^{in}_{H_R}(S)|$. If $r \notin S$, then

$$|\delta^{in}_{H_R}(S)| = |\delta^{in}_H(V\backslash S)| \geq f(V\backslash S) = f^{in}(S).$$

Thus, H is f^{out}-connected and H_R is f^{in}-connected, if H is f-connected.

Now we show the reverse implication. Again let S be any non-empty proper subset of V. We show that $|\delta_H^{in}(S)| \geq f(S)$ assuming that H is f^{out}-connected and H_R is f^{in}-connected. We consider two cases, when $r \notin S$ and when $r \in S$.

When $r \notin S$:

$$|\delta_H^{in}(S)| \geq f^{out}(S) = f(S).$$

The inequality above holds because the assumption that H is f^{out}-connected and the equality comes from the definition of f^{out}.

When $r \in S$:

$$|\delta_H^{in}(S)| = |\delta_{H_R}^{in}(V \backslash S)| \geq f^{in}(V \backslash S) = f(S). \tag{5.1}$$

The inequality above holds because the assumption that H_R is f^{in}-connected and the equality comes from the definition of f^{in}.

Thus, we can conclude that graph H is f-connected, if and only if, H is f^{out}-connected and H_R is f^{in}-connected. □

5.1 DWDCN with weighted out- or in-degree constraints

We begin with the following lemma. Since the proof of this lemma is not explicitly given in [52], we provide it here for completeness. Note that the proof is similar to Theorem 4.1 in [2].

Lemma 5.2. *If there are polynomial time (α', g') and (α'', g'')-approximation algorithms for the* DWDCN *problem with (weighted) out-degree constraints and (weighted) in-degree constraints under the intersecting connectivity requirements, respectively, then there exists a polynomial time (α, g)-approximation algorithm for the* DWDCN *problem with (weighted) out-degree constraints under the crossing supermodular connectivity requirements, where $\alpha = \alpha' + \alpha''$ and $g = g' + g''$. The same ratio applies to the* DWDCN *problem with (weighted) in-degree constraints.*

Proof. Let f be a crossing supermodular set function on V and let $r \in V$ be an arbitrary fixed node. Let f^{out} and f^{in} be the intersecting supermodular functions defined in Lemma 5.1, so, $f^{out}(S) = f(S)$, if $S \in \{A \subset V : r \notin A\}$, and $f^{out}(S) = 0$ otherwise, and $f^{in}(S) = f(V \backslash S)$, if $S \in \{A \subset V : r \notin A\}$, and $f^{in}(S) = 0$ otherwise.

Let graph $G = (V, E, c, w)$, a subset $B \subseteq V$, out-degree bounds b, and the crossing supermodular function f be together the instance of the DWDCN problem with (weighted) out-degree constraints.

Consider the DWDCN problem with (weighted) out-degree constraints on graph $G = (V, E, c, w)$ with a subset $B \subseteq V$, out-degree bounds b and the intersecting supermodular function f^{out}. Let H^{out}_{opt} be an optimal solution to this problem, i.e, it is an optimal-cost f^{out}-connected subgraph of G, which satisfies the (weighted) out-degree constraints for all nodes $v \in B$. We apply the (α', g')-approximation algorithm to the problem. Then, the algorithm will return $(\alpha',\ g')$-approximation solution $H^{out} = (V, F')$, i.e, cost of H^{out} is at most α' times $c(H^{out}_{opt})$ and for each node $v \in V$, $w(\delta^{out}_{F'}(v)) \leq g'(b(v))$.

Next consider the DWDCN problem with (weighted) in-degree constraints on the reverse graph G_R of $G = (V, E, c, w)$ with a subset $B^{in} = B$, in-degree bound $b^{in} = b$ and the intersecting supermodular function f^{in}. The reverse graph G_R is defined as (V, E_R, c, w), where $E_R = \{(u, v) : (v, u) \in E\}$, $c(u, v) = c(v, u)$, and $w(u, v) = w(v, u)$. Let H^{in}_{opt} be an optimal solution to this problem, i.e, it is an optimal-cost f^{in}-connected subgraph of G_R, which satisfies the (weighted) in-degree constraints for all nodes v. We apply the (α'', g'')-approximation algorithm to the problem. Then, the algorithm will return $(\alpha'',\ g'')$-approximation solution $H^{in} = (V, F'')$, i.e, cost of H^{in} is at most α'' times $c(H^{in}_{opt})$ and for each node $v \in V$, $w(\delta^{in}_{F''}(v)) \leq g''(b(v))$.

As the approximate solution to the DWDCN problem with (weighted) out-degree constraints under the crossing supermodular connectivity requirements we take the subgraph $H = H^{out} \bigcup H^{in}_R = (V, F)$, where $H^{in}_R = (V, F''_R)$ is the reverse graph of H^{in} and $F = F' \cup F''_R$. Since H is f^{out}-connected graph (as a super-graph of H^{out}) and the reverse graph of H is f^{in}-connected (as a super-graph of H^{in}), Lemma 5.1 implies that the graph H is f-connected.

Let H_{opt} be the optimal solution to the DWDCN problem with (weighted) out-degree constraints under the crossing supermodular connectivity requirements. We get the claimed approximation ratio for the cost since

$$\begin{aligned} c(H) &= c(F) = c(F' \cup F'') \leq c(F') + c(F'') \leq \alpha' c(H^{in}_{opt}) + \alpha'' c(H^{out}_{opt}) \\ &\leq \alpha' c(H_{opt}) + \alpha'' c(H_{opt}) = (\alpha' + \alpha'') \cdot c(H_{opt}). \end{aligned}$$

For the last inequality, observe that $c(H^{out}_{opt}) \leq c(H_{opt})$ because $f^{out}(S) \leq f(S)$ for each $S \subseteq V$, so each f-connected subgraph satisfying the out-degree constraints is also f^{out}-connected subgraph satisfying the out-degree constraints. Similarly, $c(H^{in}_{opt}) \leq c(H_{opt})$, because for each f-connected subgraph H of G satisfying out-degree constraints of H_R is a f^{in}-connected subgraph G_R satisfying the in-degree constraints.

The graph H also satisfies the claimed weighted degree bounds since

$$\begin{aligned} w(\delta^{out}_H(v)) &= w(\delta^{out}_F(v)) = w(\delta^{out}_{F' \cup F''_R}(v)) \leq w(\delta^{out}_{F'}(v)) + w(\delta^{out}_{F''_R}(v)) \\ &= w(\delta^{out}_{F'}(v)) + w(\delta^{in}_{F''}(v)) \leq g'(b(v)) + g''(b(v)) = g(b(v)). \end{aligned}$$

Let graph $G = (V, E, c, w)$, a subset $B^{in} \subseteq V$, in-degree bounds b^{in}, and the crossing supermodular function f be the instance of the DWDCN problem with (weighted) in-degree constraints under the crossing supermodular connectivity requirements. We can analogously show that the DWDCN problem with (weighted) in-degree constraints also admits the same approximation bounds. In this case, we apply (α'', β'')-approximation algorithm for the DWDCN problem with (weighted) in-degree constraints to the instance $\langle G = (V, E, c, w), B^{in}, b^{in}, f^{out} \rangle$ and apply (α', β')-approximation algorithm for the DWDCN problem with (weighted) out-degree constraints to the instance $\langle G_R = (V, E_R, c, w), B = B^{in}, b = b^{in}, f^{in} \rangle$. □

Nutov [52] has shown that under intersecting connectivity requirements there is $(1, \min\{4, f_{\max}\} \cdot b^{in}(v)\})$-approximation algorithm for the DWDCN problem with weighted in-degree constraints, where $f_{\max} = \max_{S \subseteq V} f(S)$. For the case of unit weight, he also showed that the problem admits a polynomial time $(1,\ \min\{f_{\max}, b^{in}(v)\}$-approximation algorithm.

We are now ready to prove Theorem 3.3. Theorem 3.3 is implied by Lemma 5.2, Theorem 3.1, 3.2 and above result of Nutov. For example, $(4,\ (5+\min\{3, f_{\max}\}) \cdot b(v))$-approximation bounds for the DWDCN problem with weighted out-degree (in-degree) constraints under crossing supermodular function f is obtained by the $(2,\ 5b(v))$-approximation algorithm given in Theorem 3.1 and the $(2,\ \min\{3, f_{\max}\} \cdot b^{in}(v))$-approximation algorithm given in Theorem 3.2.

5.2 DWDCN with both weighted out- and in-degree constraints

Here we consider the case of both out-degree and in-degree constraints. Lemma 5.2 can be extended to deal with the case when there are both out-degree and in-degree constraints. Again the proof of the following extension of Lemma 5.2 is not explicitly given in [52], we provide it here for completeness.

Lemma 5.3. *There is a polynomial time* $(2\alpha,\ g,\ g)$*-approximation algorithm for the* DWDCN *problem with (weighted) out- and in-degree constraints under the crossing supermodular connectivity requirements if there is a polynomial time* $(\alpha,\ g',\ g'')$*-approximation algorithm for the same problem under the intersecting supermodular connectivity requirements and* $g' + g'' = g$.

Proof. Let f be a crossing supermodular set function on V and let $r \in V$ be an arbitrary fixed node. Let f^{out} and f^{in} be the intersecting supermodular functions defined in Lemma 5.1, so,

$f^{out}(S) = f(S)$, if $S \in \{A \subset V : r \notin A\}$, and $f^{out}(S) = 0$ otherwise, and $f^{in}(S) = f(V \backslash S)$, if $S \in \{A \subset V : r \notin A\}$, and $f^{in}(S) = 0$ otherwise.

Consider the DWDCN problem with (weighted) out- and in-degree constraints on graph $G = (V, E, c, w)$ with a subset $B \subseteq V$, out-degree upper bounds $b = \{b(v) : v \in B \subseteq V\}$, a subset $B^{in} \subseteq V$ and in-degree upper bounds $b^{in} = \{b^{in}(v) : v \in B^{in} \subseteq V\}$ and the intersecting supermodular function f^{out}. Let H^{out}_{opt} be the optimal solution to the this problem, i.e, it is an optimal cost f^{out}-connected subgraph of G, which satisfies both (weighted) out- and in-degree constraints for all nodes v. We apply the (α, g', g'')-approximation algorithm to the problem. Then, it will return $(\alpha,\ g',\ g'')$-approximation solution $H^{out} = (V, F')$, i.e, cost of H^{out} is α times $c(H^{out}_{opt})$, for each node $v \in B$, $w(\delta^{out}_{F'}(v)) \leq g'(b(v))$ and for each node $v \in B^{in}$, $w(\delta^{in}_{F'}(v)) \leq g''(b^{in}(v))$.

Next consider the DWDCN problem with (weighted) out- and in-degree constraints on the reverse graph G_R of $G = (V, E, c, w)$ with a subset $B = B^{in}$, out-degree upper bound $b(v) = b^{in}(v)$, a subset $B^{in} = B$, in-degree upper bounds $b^{in}(v) = b(v)$ and the intersecting supermodular function f^{in}. Again the reverse graph of G_R is defined as (V, E_R, c, w), where $E_R = \{(u, v) : (v, u) \in E\}$, $c(u, v) = c(v, u)$, and $w(u, v) = w(v, u)$. Let H^{in}_{opt} be the optimal solution to this problem, i.e, optimal cost f^{in}-connected subgraph of G_R, which satisfies both (weighted) out- and in-degree constraints for all nodes v. We apply the (α, g', g'')-approximation algorithm to the problem. Then, it will return $(\alpha,\ g',\ g'')$-approximation solution $H^{in} = (V, F'')$ of G_R, i.e, cost of H^{in} is α times $c(H^{in}_{opt})$ and for each node $v \in B$, $w(\delta^{out}_{F''}(v)) \leq g'(b^{in}(v))$ and for each node $v \in B^{in}$, $w(\delta^{in}_{F''}(v)) \leq g''(b(v))$.

The solution to the DWDCN problem with (weighted) out- and in-degree constraints under the crossing supermodular connectivity requirements is graph $H = H^{out} \bigcup H^{in}_R = (V, F)$, where $F = F' \cup F''_R$. Since H is f^{out}-connected graph and reverse graph of H is f^{in}-connected, Lemma 5.1 implies that the graph H is f-connected.

Let H_{opt} be the optimal solution to the DWDCN problem with (weighted) out-degree constraints under the crossing supermodular connectivity requirements. We get the claimed

approximation ratio for the cost since

$$\begin{aligned} c(H) &= c(F) = c(F' \cup F'') \leq c(F') + c(F'') \leq \alpha \cdot c(H^{in}_{opt}) + \alpha \cdot c(H^{out}_{opt}) \\ &\leq \alpha \cdot c(H_{opt}) + \alpha \cdot c(H_{opt}) = 2\alpha \cdot c(H_{opt}). \end{aligned}$$

The graph H also satisfies the claimed (weighted) out-degree bounds since

$$\begin{aligned} w(\delta^{out}_H(v)) &= w(\delta^{out}_F(v)) = w(\delta^{out}_{F' \cup F''_R}(v)) \leq w(\delta^{out}_{F'}(v)) + w(\delta^{out}_{F''_R}(v)) \\ &= w(\delta^{out}_{F'}(v)) + w(\delta^{in}_{F''}(v)) \leq g'(b(v)) + g''(b(v)) = g(b(v)). \end{aligned}$$

Moreover, the graph H also satisfies the claimed (weighted) in-degree bounds since

$$\begin{aligned} w(\delta^{in}_H(v)) &= w(\delta^{in}_F(v)) = w(\delta^{out}_{F' \cup F''_R}(v)) \leq w(\delta^{in}_{F'}(v)) + w(\delta^{in}_{F''_R}(v)) \\ &= w(\delta^{in}_{F'}(v)) + w(\delta^{out}_{F''}(v)) \leq g''(b^{in}(v)) + g'(b^{in}(v)) = g(b^{in}(v)). \end{aligned}$$

□

Theorem 3.4 and Lemma 5.3 implies Theorem 3.5.

Chapter 6

Maximum Network Lifetime (MNL) problems

In Chapters 3–5, we discussed the Directed Weighted Degree Constrained Network Design (DWDCN) problems and developed solutions, which improve a number of previous approximation bounds for these problems. The algorithms for DWDCN can be applied to solve the *Maximum Network Lifetime* (MNL) problems. We discuss in Chapters 6 – 8 how this can be done and show how new improved approximation bounds for MNL problems can be obtained.

In an MNL problem, we are given a communication task and wireless ad-hoc network $N = (V, E, w, B)$, where w is an edge-weight function representing the energy costs of individual transmissions and B is a node-battery function representing the initial battery capacity of nodes. The goal is to compute a maximum-size collection of *routing topologies* (schedules of individual transmissions) for the specified communication task, which satisfy the *energy constraints*. That is, every node must have sufficient battery capacity to support the specified communication task over all communication rounds, where each communication round is defined by one routing topology from the computed collection.

The MNL problems can be categorised into *single topology* and *multiple topology* problems [57], which are distinguished by different output requirements. In the single topology variant,

the same routing topology is used in all communication rounds. For example, the same broadcast tree is used in each round to broadcast from a given root node r. In the multiple topology variant, the routing topologies can be different in different rounds. That is, the outcome for a single topology MNL problem is one routing topology, while the outcome of a multiple topology MNL problem is a collection of routing topologies.

We consider both single and multiple topology MNL problems. Thus in this thesis we refer to the following eight problems (the acronyms are in the brackets):

- Single Topology and Multiple Topology Unicast (STUand MTU),
- Single Topology and Multiple Topology Broadcast (STBand MTB),
- Single Topology and Multiple Topology Convergecast (STCand MTC),
- Single Topology and Multiple Topology Mixedcast (STMand MTM).

In Section 6.1, we discuss the models of wireless ad-hoc networks that are commonly used in the literature. In Section 6.2, we describe network model, used in this thesis and provide basic definitions related to the communication tasks and routing topologies. In Section 6.3, we formally define the MNL problems, which we consider. In Sections 6.4 and 6.5, we discuss previous related results and summarise our contributions to the MNL problems. In Section 6.6, we discuss the computational complexity of the MNL problems, and extend previous NP-hardness proofs of the MNL problems to some variants of these problems considered in this thesis. In Chapter 7, we describe the approximation algorithms for the MNL unicast, broadcast and convergecast problems and derive new improved approximation bounds for these problems are obtained. In Chapter 8, we consider the MNL mixedcast problems.

6.1 Various models of wireless ad-hoc networks

Numerous algorithms and heuristics have been proposed to maximise the network lifetime in wireless ad-hoc networks. However, it is difficult to compare these algorithms to one another as the models employed to describe the wireless ad-hoc networks vary depending on some factors such as the types of antenna used for communication and the assumptions made for the environment settings.

Typical considerations in modelling the behaviour of wireless ad-hoc networks are:

- undirected (that is, symmetric) communication versus directed (that is, asymmetric) communication,
- omnidirectional antenna versus directional antenna,
- adjustable transmission power level versus fixed transmission power level,
- data aggregation versus no data aggregation,
- channel access methods and interference.

In the *undirected model*, also known as the *symmetric model*, there is an undirected communication link between two nodes, which means that if a node u can reach a node v, the node v can also reach node u and the communication costs in both directions are the same. On the other hand, the *directed model*, also known as *asymmetric model*, specifies one way communication links between two nodes. Generally, the "conventional" network (the wired network) algorithms (or protocols) are developed under the undirected model. However, in wireless ad-hoc networks as communications can be easily be affected by background noise and interference that are likely to be non-uniform, the directed model is often adopted to develop algorithms (or protocols). The directed model seems to reflect the more realistic model of wireless ad-hoc networks. The directional model has been considered in [13, 28, 29, 44, 54, 57, 61, 63], whereas, the undirected model has been considered in [5, 45, 67, 70].

Another important factor for modeling the wireless ad-hoc networks is the type of antennas used by nodes for communication. In *omnidirectional* antenna model, considered, for example, in [5, 13, 28, 29, 44, 45, 57, 61, 63, 67, 69, 70, 15], all nodes have a 360 degree coverage. This means that, for instance, if a node u transmits a message with enough power to reach another node at distance d, then all nodes, which are located within the distance d in any direction, are also able to receive the message. On the other hand, in directional antenna model, each node has a limited angular coverage (the angle is defined by the angle of the beam width), hence, only the set of nodes, which are located within distance d in that direction, are able to receive the message. In the extreme case of the directional antenna model, each transmission has a single-recepient. This model is often referred to as the unidirectional antenna model, and was considered in [13, 54, 57, 61].

Another consideration in terms of the network model of wireless ad-hoc network is the transmission power level whether it is adjustable by nodes or not. The transmission power level of a node determines the transmission range of the node and it affects the performance of the network in various aspects. For instance, if transmission power level of a node is chosen to be high, then the transmission range of this node increases, and consequently the number of intermediate nodes needed to reach the intended destination decreases. This can reduce the traffic in the network, but may undesirably increase the magnitude of interference.

There are different ways of using adjustable transmission power level in the context of maximising network lifetime in wireless ad-hoc networks. For example, in [44, 63, 45] transmission power level of a node v is initially set to the maximum, so that each neighbour within that range can receive the message from v. Once the intended target node is set, each node can dynamically reduce the transmission power to the level which is just enough to reach the target. This may help to improve the energy usage in the network. In [20, 70], the authors assume that all nodes transmit at the same fixed power level. Gomez *et al.* in [24] show that adjustable transmission range can improve the overall network performance.

Data aggregation is a process of aggregating the obtained data into smaller size message to enhance the energy efficiency. This is important in data gathering (convergecast) communi-

cation tasks. Some models allow a perfect (full) aggregation [44, 45, 54, 70, 29, 61], which means that multiple incoming messages of size x are compressed into a single message of size x. Such data gathering scenarios include simple database queries such as MAX, MIN, COUNT, SUM, and AVERAGE. The perfect aggregation is ideal in terms of energy efficiency, since it is natural to assume that the energy required to transmit a message is proportional to the size of this message. However, data aggregation is not always applicable. In certain data gathering scenarios, such as gathering video of the local area, the received messages from different part of the area may not be aggregated into the same size in any meaningful way. Thus, there have been works on different types of models with data aggregation, full (see references as above), or partial [7], and without data aggregation [29]. Li *et al.* in [42] analyse the trade-off between communication delay and energy consumption of all three types of data aggregation.

An additional consideration for wireless ad-hoc network model is a channel access method. Wireless communication uses radio frequencies (RF) to transmit a message, and a collision of transmissions may occur if two or more nodes transmit a message using the same frequency at the same time at the same place. As a consequence, the intended receivers may not receive the original messages. In order to support multiple sessions simultaneously without having a collision, a channel access method is used. The media access control (MAC) layer protocols address such issues. There are several well-known methods, such as frequency division multiple access (FDMA), code division multiple access (CDMA), time division multiple access (TDMA), and many variants of them. The majority of papers, which address the maximum network lifetime problem, do not consider scheduling of transmissions. Instead, they assume that MAC layer uses "perfect" TDMA-based or CDMA-based scheduling, so that collision and interference do not occur.

6.2 Our model and preliminaries

In this section, we describe our network model (Section 6.2.1) and give formal definitions of communication tasks and routing topologies (Section 6.2.2).

6.2.1 Our Model

We consider a wireless ad-hoc network N consisting of n stationary nodes. Each node v is equipped with a unidirectional antenna, which only permits a single node to receive a transmitted message at a time. Each node v has a finite amount of initial battery capacity and each node-to-node transmission uses a fixed (given) amount of energy at the transmitting node. We proceed with a formal definition. Recall that $\mathbb{R}^+$ is the set of non-negative real numbers.

Definition 6. *A static wireless ad-hoc network $N = (V, E, w, B)$ is modelled as a weighted, directed graph (V, E), where V is a set of nodes with $|V| = n$, $E \subseteq V \times V$ is a set of directed edges, $w : E \to \mathbb{R}^+$ is an edge-weight function representing energy cost of transmissions, and $B : V \to \mathbb{R}^+$ is a battery capacity function.*

In the network N, a directed edge (u, v) indicates that node u is able to directly transmit a message to node v. An edge-weight $w(u, v)$ of the directed edge (u, v) denotes the amount of energy consumed to transmit one message from node u to node v. For example, if the network is embedded into a physical space, then we could consider $w(u, v) = d(u, v)^\alpha$, where $d(u, v)$ is distance from node u to v and α is a path attenuation factor, usually taken to be between 2 and 4. However, in our model, we do not assume any particular relation between the edge-weight and the distance between the nodes in the physical space. The edge weights are simply part of the input. The battery capacity $B(v)$ denotes the initial battery capacity of node v. To support the heterogeneity of nodes in the network, we allow the initial battery capacities to be different.

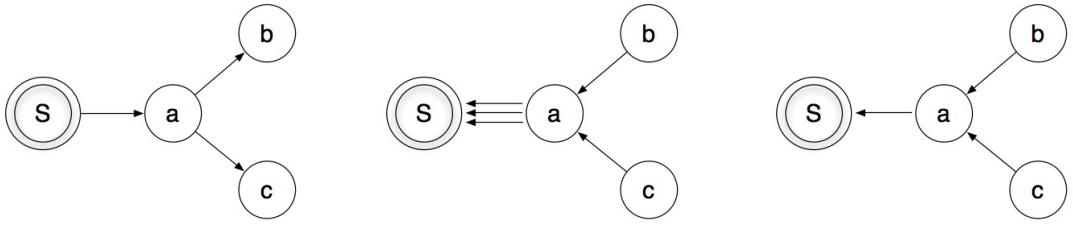
In our model, we take into account only the energy consumption of transmissions, assuming that in wireless networks the radio frequency transmission dominates the energy usage. In particular, we do not consider energy consumption for receiving and processing data. We note that some previous works also consider the energy consumption for receiving [26, 72, 70, 45, 10]. We assume that every node shares the same frequency band and the MAC layer is based on "collision-free" TDMA, so that transmissions do not interfere with each other. For the convergecast problem, we assume that the messages from different nodes can always be fully

aggregated into one message. This means that if we want to minimise the energy used by a node, then this node should wait for all messages it is to receive in the current round, and then aggregate all of them into one message and send it on towards the root.

Communication in a wireless network is *fractional* or *discrete*. In the fractional variant [29, 57], a message is allowed to be divided into smaller messages, which can be transmitted separately and possibly along different routes. Whereas, in the discrete variant [13, 54, 58, 6], each message has to be sent in one transmission. The discrete variant seems to reflect better the existing network protocols, because in the variant messages are regarded as the smallest data unit (e.g, packet), which cannot be further split. We consider the discrete variant in this thesis.

6.2.2 Communication tasks and routing topologies

In wireless ad-hoc networks, *unicasting*, *broadcasting*, and *convergecasting* are the fundamental tasks in network communication. Many applications as mentioned earlier are based on these communication tasks. Unicasting (or unicast) is one-to-one communication, where information held in one node (called the source) in the network is transmitted to another node (called the destination). Broadcasting (or broadcast) is one-to-all communication, where information held in one node (the source) is transmitted to all other nodes. Convergecasting (or convergecast or *data gathering*) can be viewed as the opposite to broadcast, when data from all nodes are transmitted to one specified node (the sink or destination).



(a) (b) (c)

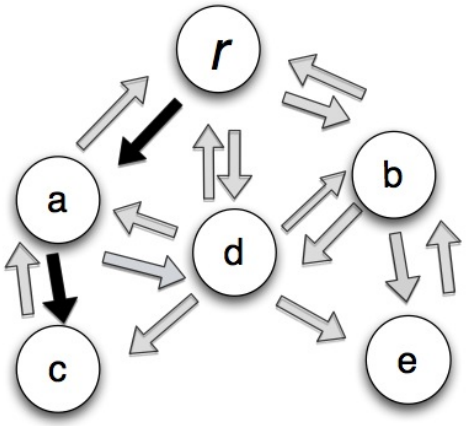
Figure 6.1 (a) Broadcast - data flows from a single node s to all nodes a, b, and c. (b) Convergecast without aggregation - data flows from nodes a, b, and c to a single node s. (c) Convergecast with aggregation.

Figure 6.1 shows a simple example that illustrates the characteristics of broadcast and convergecast. Each arrow corresponds to a single node-to-node transmission. In a broadcast, as shown in Figure 6.1(a), node s is the source and all other nodes a, b, and c are its expected recipients. Node a received the message directly from node s and relayed the message to nodes b and c. In convergecast, illustrated in Figure 6.1(b), each node a, b, and c has a message intended for the sink (destination) s, and a serves as a relay node for nodes b and c. The three arrows indicates that there is no aggregation at node a, and three independent transmission have to be made. A single arrow from a to s in Figure 6.1(c) indicates that a performs full data aggregation.

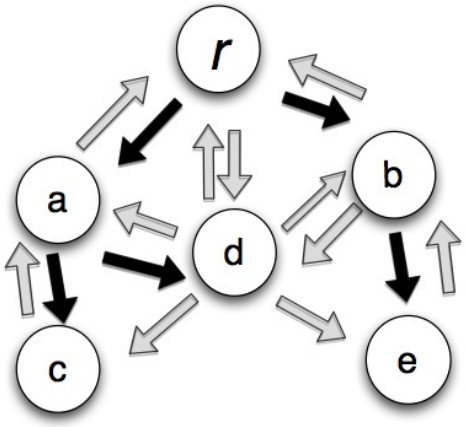
A *routing topology* defines an execution of a given communication task. We refer to one execution of a communication task as one communication round. The structure of a routing topology R changes depending on the type of the communication task. For broadcast and convergecast, the properties of our model imply that optimal solutions use tree-based routing topologies. Thus, a routing topology R is in the form of a broadcast tree T_{out} or convergecast tree T_{in}, respectively, with the root node, the source of the broadcast or the destination of the convergecast. A routing topology for unicast is a simple path P from the source node r to the destination node s. Figure 6.2 shows examples of a structure of routing topology for unicast, broadcast, and convergecast. In each round a specified communication task is executed based upon the information of the pre-computed routing topology.

Accordingly, a routing topology for a given communication task refers to the corresponding structure, a path P for unicast, a tree T_{in} for convergecast, and a tree T_{out} for broadcast.

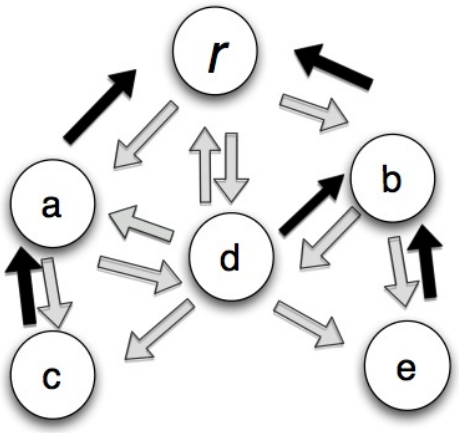
As mentioned in the beginning of this chapter, we consider both single topology and multiple topology cases. In the single topology case, a routing topology is used for all communication rounds. On the other hand, in the multiple topology case, the routing topologies used in different rounds do not have to be identical. The single topology case is easier to deploy, but multiple topology case can be more energy efficient.



(a) Unicast: A path from source r to destination c



(b) Broadcast: a tree T_{out} rooted at source r



(c) Convergecast: a tree T_{in}

Figure 6.2 A structure of routing topology for unicast, broadcast, and convergecast

6.3 Definitions of the problems

The *Maximum Network Lifetime* (MNL) problem for unicast, broadcast and convergecast is defined as follows. The input to the problems is a network $N = (V, E, w, B)$, and a node $r \in V$ (for broadcast and convergecast) or two nodes $r, s \in V$ (for unicast). The output is a collection of routing topologies $\mathscr{R} = \{R_1, ..., R_k\}$ for the given communication task, which satisfy the

following energy constraints:

$$\sum_{i=1}^{k} w(\delta_{R_i}^{out}(v)) \;=\; \sum_{i=1}^{k} \sum \{w(e) : e \in \delta_{R_i}^{out}(v)\} \;\leq\; B(v), \quad \text{for all} \quad v \in V. \tag{6.1}$$

The left-hand side of (6.1) is the total energy used by node v over k communication rounds, when the i^{th} round is done according to the routing topology R_i. The optimisation objective of the problem is to maximise k. Since k represents the number of communication rounds, which can be executed within the specified battery capacities, larger k means longer network lifetime.

In the single topology variant, the same routing topology R is employed for all k rounds, i.e. $R_i = R$ for all $i \leq k$. In this case, the constraints (6.1) simplify to the following constraints:

$$k \cdot \sum \{w(e) : e \in \delta_{R}^{out}(v)\} \;\leq\; B(v), \quad \text{for all} \quad v \in V. \tag{6.2}$$

The MNL *mixedcast* problem is defined in the following way. We are given two positive integer parameters τ and γ, and the objective is to find the maximum integer k such that τk broadcasts and γk convergecasts can be performed whilst the energy constraints are satisfied. More formally, the input to this problem is a network $N = (V, E, w, B)$, two nodes $r_b, r_c \in V$, and two integers $\tau, \gamma \geq 1$. The output is a maximum integer k, a collection of τk broadcast trees $\mathscr{T}_{out} = \{T'_1, ..., T'_{\tau k}\}$ rooted at node r_b and a collection of γk convergecast trees $\mathscr{T}_{in} = \{T''_1, ..., T''_{\gamma k}\}$ rooted at node r_c, that satisfy the following energy constraints:

$$\sum_{i=1}^{\tau k} \sum \{w(e) : e \in \delta_{T'_i}^{out}(v)\} \;+\; \sum_{i=1}^{\gamma k} \sum \{w(e) : e \in \delta_{T''_i}^{out}(v)\} \;\leq\; B(v), \quad \text{for all} \quad v \in V. \tag{6.3}$$

In the single topology variant, we need to find one broadcast tree T_{out} and one convergecast tree T_{in} that are feasible for τk broadcast rounds and γk convergecast rounds. Hence, the energy constraints (6.3) simplify to:

$$\tau k \cdot \sum \{w(e) : e \in \delta_{T_{out}}^{out}(v)\} \;+\; \gamma k \cdot \sum \{w(e) : e \in \delta_{T_{in}}^{out}(v)\} \;\leq\; B(v), \quad \text{for all} \quad v \in V. \tag{6.4}$$

Thus, in this thesis we consider the following *eight* MNL problems.

Single Topology Unicast (STU) *problem*:

Input: a network $N = (V, E, w, B)$, and two nodes $r, s \in V$.

Output: a maximum integer k and an r-s path P that satisfy the energy constraints (6.2).

Multiple Topology Unicast (MTU) *problem*:

Input: a network $N = (V, E, w, B)$, and two nodes $r, s \in V$.

Output: a maximum-size collection of r-s paths $\mathscr{P} = \{P_1, ..., P_k\}$ that satisfies the energy constraints (6.1).

Single Topology Broadcast (STB) *problem*:

Input: a network $N = (V, E, w, B)$, and node $r \in V$.

Output: a maximum integer k and a broadcast tree T_{out} rooted at r that satisfy the energy constraints (6.2).

Multiple Topology Broadcast (MTB) *problem*:

Input: a network $N = (V, E, w, B)$, and node $r \in V$.

Output: a maximum-size collection of broadcast trees $\mathscr{T}_{out} = \{T_1, ..., T_k\}$ rooted at r that satisfies the energy constraints (6.1).

Single Topology Convergecast (STC) *problem*:

Input: a network $N = (V, E, w, B)$, and node $r \in V$.

Output: a maximum integer k and a convergecast tree T_{in} rooted at r that satisfy the energy constraints (6.2).

Multiple Topology Convergecast (MTC) *problem*:

Input: a network $N = (V, E, w, B)$, and node $r \in V$.

Output: a maximum-size collection of convergecast trees $\mathscr{T}_{in} = \{T_1, ..., T_k\}$ rooted at r that satisfies the energy constraints (6.1).

Single Topology Mixedcast (STM) *problem*:

Input: a network $N = (V, E, w, B)$, two nodes $r_c, r_b \in V$, and two integers $\tau, \gamma \geq 0$.

Output: a maximum integer k, a broadcast tree T_{out} rooted at node r_b, and a convergecast T_{in} rooted at node r_c, which together satisfy the following energy constraints (6.4).

Multiple Topology Mixedcast (MTM) *problem*:

Input: a network $N = (V, E, w, B)$, two nodes $r_c, r_b \in V$, and two integers $\tau, \gamma \geq 0$.

Output: a maximum integer k, a collection of broadcast trees $\mathscr{T}_{out} = \{T_1, ..., T_{\tau k}\}$ rooted at node r_b and a collection of convergecast trees $\mathscr{T}_{in} = \{T_1, ..., T_{\gamma k}\}$ rooted at node r_c, which satisfy the energy constraints (6.3).

6.4 Previous Results

The previous studies of the topic of maximising the lifetime of ad-hoc wireless networks have been considering mainly the omnidirectional communication model [30, 44, 45, 57, 58, 70]. Kang and Poovendran [30] investigate the fractional variants of the Maximum Network Lifetime (MNL) problem for broadcast communication, proposing a polynomial time algorithm for the STB problem and some heuristics for the MTB problem. Orda and Yassour [57] improve the time complexity of the STB and STU problem, prove that the MTB problem is NP-hard, and propose additional MTB heuristics. Segal [61] further improves the running time of the STB problem showing that an optimal solution can be computed in $O(|V| + |E|)$. In the same paper, Segal shows that the STC problem can also be solved in linear time, showing the same approach as for the STB problem. Additional results related to the maximum network lifetime problem under broadcast communication can be found in [11, 47, 58].

Kalpakis *et al.* [29] consider the fractional variants of the MTC problem with full aggregation, giving a polynomial time algorithm, but their polynomial bound is of high-degree. For the same problem, Stanford and Tongngam [64] give $(1-\varepsilon)$-approximation algorithm with a

considerably faster running time. Dasgupta *et al.* [10] also propose and experimentally evaluate a heuristic for this problem.

There have been many studies related to the convergecast problem with an objective of maximizing the lifetime of the network under the various assumptions on the network structures and energy model. Wu *et al.* [70] consider the problem assuming uniform transmission costs. The model with arbitrary transmission costs is considered in [45]. In [44, 45, 50], only the energy consumption of transmissions is taken into account. In [29, 65, 70] the authors consider more general model, in which the energy consumption of receiving messages is also taken into account. The alternative approach of optimising only a single session has been considered in [43, 69, 71] for broadcast and in [22, 33, 66] for convergecast.

Orda and Yassour [57] were the first to consider the complexity of the MNL problem in *unidirectional* communication model. Under this model, they show that the fractional variant of the STB problem is NP-hard, and propose a polynomial time algorithm for the fractional variant of the MTB problem. It is not difficult to show that in this model the STU and STC problems can be solved in polynomial time. Segal [61] shows that we can actually get a linear time algorithm for this problem. Bodlaender *et al.* [6] show that the decision variant of the multiple topology unicast (MTU) problem is *strongly* NP-complete and APX-hard. A simple reduction from MTU to MTC implies that MTC is also strongly NP-complete and APX-hard. Elkin *et al.* [13] show that the broadcast problems STB and MTB are NP-hard. Actually, the special case of the MNL broadcast problems which asks whether a given network allows one broadcast is already NP-complete (a simple reduction from the Hamiltonian path problem).

Elkin *et al.* [13] give also an $\Omega(1/\log n)$-approximation algorithm for the STB problem, under the assumption that k_{opt} (the maximal number of rounds) is appropriately large. Nutov [51] shows a constant-ratio approximation algorithm for the MTU problem, and Nutov and Segal [54] show constant-ratio approximation algorithms for the STB, MTB and MTC problems, if k_{opt} is appropriately large. They also show that the MTC problem admits a $1/31$-approximation polynomial time algorithm. The previous best results for the MNL problems are given in

Table 6.1. Note that the results in the table guarantee the number of rounds $\lfloor k_{opt}/\beta \rfloor$ only for the inputs such that $w(u,v) \leq B(u)/\beta$, for each edge (u,v).

Table 6.1 Previous results and our results for the MNL problems. The previous results are due to Nutov [51] and Nutov and Segal [54].

Previous results				
Type of solution	unicast	convergecast	broadcast	mixedcast
Single Topology	k_{opt}	k_{opt}	$\lfloor k_{opt}/25 \rfloor$ [54]	$\lfloor k_{opt}/36 \rfloor$ [54]
Multiple Topology	$\lfloor k_{opt}/16 \rfloor$ [51] $1/31 \cdot k_{opt}$ [51]	$\lfloor k_{opt}/16 \rfloor$ [54] $1/31 \cdot k_{opt}$ [54]	$\lfloor k_{opt}/36 \rfloor$ [54]	$\lfloor k_{opt}/100 \rfloor$ [54]
Our results				
Type of solution	unicast	convergecast	broadcast	mixedcast
Single Topology	-	-	$\lfloor k_{opt}/5 \rfloor$	$\lfloor k_{opt}/5 \rfloor$
Multiple Topology	$\lfloor k_{opt}/3 \rfloor$ $1/5 \cdot k_{opt}$	$\lfloor k_{opt}/3 \rfloor$ $1/5 \cdot k_{opt}$	$\lfloor k_{opt}/5 \rfloor$	$\lfloor k_{opt}/5 \rfloor$

6.5 Our Results

We present the algorithmic results for the discrete variant of the *Maximum Network Lifetime* (MNL) problems for unicast, broadcast, convergecast, and mixedcast, under the unidirectional antenna model. We consider both the single and multiple topology variants of these problems.

Our results are based on the Maximum Network Lifetime (MNL) approximation algorithms proposed by Nutov and Segal's [54], which find a good value of k using binary search and Nutov's bicriteria algorithm for Directed Weighted Degree Constrained Network Design DWDCN problems [50]. We give a different analysis of the overall binary search and using our new improved approximation bounds for the DWDCN problems, which is given in Chapters 3–4, we obtain better approximation factors for unicast, convergecast, and broadcast problems than the ones obtained in [51, 54]. These improved approximation factors, together with a new approach to mixedcast problem, give our new approximation factors for the mixedcast.

The following theorem gives the approximation factors for the MTU, STB, MTB, and MTC problems.

Theorem 6.1. *For each of the problems,* MTU, STB, MTB, *and* MTC, *there exists a polynomial time algorithm, which finds a solution with* $k \geq \lfloor k_{opt}/\beta \rfloor$ *where:*

- $\beta = 3$, *for the* MTU *and* MTC *problems;*
- $\beta = 5$, *for the* STB *and* MTB *problems;*

for all input instances of STB and for all input instances of MTU, MTB and MTC such that $w(u,v) \leq B(u)/\beta$ *for each edge* (u,v).

We note that our results guarantee the number of rounds $\lfloor k_{opt}/\beta \rfloor$ for the MTU and MTB and MTC problems only for the inputs such that $w(u,v) \leq B(v)/\beta$.

The Single Topology Convergecast (STC) problem can be solved in polynomial time [61]. Therefore, for the Multiple Topology Convergecast (MTC) problem, we can determine in polynomial time whether $k_{opt} \geq 1$. This yields the following corollary.

Corollary 6.2. *The Multiple Topology Convergecast (*MTC*) problem admits a* $1/5$*-approximation polynomial time algorithm, for inputs such that* $w(u,v) \leq B(v)/3$ *for each edge* (u,v).

Proof. Let k_{opt}^{STC} and k_{opt}^{MTC} denote the optimal number of rounds for the STC and MTC problems, respectively. We run a polynomial time algorithm for STC problem to obtain k_{opt}^{STC}. If $k_{opt}^{STC} = 0$, then $k_{opt}^{MTC} = 0$. If $k_{opt}^{STC} \geq 1$, then we run the polynomial time algorithm of Theorem 6.1 to get $k \geq \lfloor k_{opt}^{MTC}/3 \rfloor$ convergecast trees. Our solution for the MTC problem is now $k_{sol} = \max\{k_{opt}^{STC}, k\} \geq 1$. Because $k_{sol} \geq 1$, we have,

$$k_{sol} \geq \left\lfloor \frac{k_{opt}^{MTC}}{3} \right\rfloor \geq \frac{k_{opt}^{MTC}}{3} - \frac{2}{3} \geq \frac{k_{opt}^{MTC}}{3} - \frac{2 \cdot k_{sol}}{3}.$$

This implies that $k_{sol} \geq k_{opt}^{MTC}/5$. □

An analogous bound applies to the Multiple Topology Unicast (MTU) problem, since the Single Topology Unicast (STU) problem has a polynomial time algorithm.

Corollary 6.3. *The Multiple Topology Maximum Lifetime Unicast (*MTU*) problem admits a* $1/5$*- approximation polynomial time algorithm, for inputs such that* $w(u,v) \leq B(v)/3$.

Our results for the mixedcast problems are summarised in the following theorem.

Theorem 6.4. *For the Single and Multiple Topology Mixedcast (*STM *and* MTM*) problems, there exists a polynomial time algorithm, which finds a solution with at least* $\lfloor k_{opt}/\beta \rfloor$ *rounds where:*

- $\beta = 5$, *for the* STM *problem;*
- $\beta = 5$, *for the* MTM *problem.*

for all input instances such that $w(u,v) \geq B(v)/\beta$.

Our results for the MNL problems are summarised in Table 6.1.

6.6 Computational complexities of the MNL problems

In this section, we discuss the computational complexity of the MNL problems.

The single topology convergecast (STC) problem of finding a maximum integer k and a convergecast tree rooted at r that satisfy the energy constraints (6.2) can be solved in polynomial time. This can be done as follows. For each edge $e = (u,v)$ in G, we calculate the values $k(u,v) = \left\lfloor \frac{B(u)}{w(u,v)} \right\rfloor$, and order the edges of G into a sequence $e_1, e_2, \ldots, e_m$ according to

increasing values of $k(e_i)$. For each $i = 1,2,\ldots,m$, let $G_i = (V, \{e_i, e_{i+1}, \ldots, e_m\})$, in particular, $G_1 = G$. Let i_{max} be the maximum index such that $G_{i_{max}}$ contains an in-arborescence rooted at node r. Then, $k_{opt} = k(e_{i_{max}})$. We can find i_{max} by binary search. Clearly, this can be done in $O(m \log n)$ time, and it was shown in [61] that it can be implemented in $O(m)$ time.

The single topology unicast (STU) problem can also be solved in polynomial time in similar way, but instead of checking for existence of an in-arborescence rooted at node r in graph G_i, we need to check whether there is a simple path from source r to destination s.

It is shown in [6] that the decision variant of the multiple topology unicast (MTU) problem is *strongly* NP-complete and APX-hard, by reduction from 3-partition problem. A simple reduction from MTU to MTC shows that MTC is also strongly NP-complete and APX-hard. Since we could not find a reference to this reduction in the previous literature, we give it below for completeness.

Theorem 6.5. *The multiple topology convergecast (MTC) problem is NP-complete and APX-hard.*

Proof. Consider the following polynomial time reduction from MTU to MTC. For an input $I = \langle (V, E, w, B), r, s \rangle$ of the MTU problem, we create an instance $I' = \langle (V, E', w', B), r' \rangle$ of the MTC problem as follows. For each node $v \in V \setminus \{r, s\}$, we add a new directed edge from node v to node r with zero weight, and we set $r' = s$. It is easy to see that our instance I' of the MTC problem has k rounds, if and only if, the instance I of the MTU problem has k rounds. This is because for this instance I' it is sufficient to consider only the convergecast trees, which consist of a simple path from r to r' and the edges (v, r) from all nodes v not on this path. □

For the broadcast problems (STB and MTB), even checking whether $k = 1$ is feasible is NP-complete (simple reduction from the Hamiltonian path problem) [13]. This implies that the single and multiple topology broadcast (STB and MTB) problems are NP-hard.

Another simple reduction shows that the broadcast problems (STB and MTB) can be viewed as a special case of the mixedcast problems (STM and MTM). For an input $I = \langle (V, E, w, B), r \rangle$

of the broadcast problems, create an instance $I' = \langle (V, E', w', B), r_c, r_b, \gamma, \tau \rangle$ of the mixedcast problems as follows. We set $r_b = r_c = r$. For each node $v \in V \backslash \{r\}$, add a new directed edge from node v to node r with zero weight. We set $\tau = \gamma = 1$. Then, our instance I of the broadcast problem has k broadcast trees, if and only if, the instance I' of the mixedcast problem has a solution with k broadcast trees and k convergecast trees. Thus, the mixedcast problems are NP-hard.

The complexities of the MNL problems are depicted in Table 6.2.

Type of solution	unicast	convergecast	broadcast	mixedcast
Single Topology	P [61]	P [61]	NP-complete [13]	NP-complete
Multiple Topology	NP-complete [6]	NP-complete	NP-complete [13]	NP-complete

Table 6.2 The complexities of the Maximum Network Lifetime (MNL) problems

6.6.1 Complexities of the restricted MNL problems

Our approximation factors hold only for the inputs such that $w(u,v) \leq B(u)/\beta$, for each edge (u,v). In this section, we show that the NP-hard MNL multiple topology problems remain NP-hard for such a restricted class of inputs. The proofs are given in Theorems 6.6, 6.7 and 6.9.

Theorem 6.6. *The* MTB *problem remains NP-hard even if the input instances are restricted by the condition that for each edge* (u,v), $w(u,v) \leq B(u)/\beta$, *where* $\beta > 1$ *is fixed (but arbitrary) integer constant.*

Proof. Consider the following polynomial time reduction from the decision version of the MTB problem to the decision version of the restricted MTB problem. Let $I = \langle N = (V, E, w, B), r \in V, k \geq 1 \rangle$ be an instance of the decision version of the MTB problem: the answer for this

instance is YES, if and only if, there exists a k-round MTB solution for $\langle N, r\rangle$. We can assume that $w(u,v) \leq B(v)$ because an edge with $w(u,v) > B(u)$ is never relevant for the answer. For the instance I, we create an instance $I' = \langle N' = (V', E', w', B'), r, k \geq 1\rangle$ of the decision version of the restricted MTB problem as follows.

For each $v \in V$, we create β additional nodes $v_1, \ldots, v_\beta$. Let $V_v = \{v_1, \ldots, v_\beta\}$ be a set of β nodes created for node v. We set $V' = V \bigcup_{v \in V} V_v$. For each newly created node x associated with node v, we add a directed edge from v to x. Thus, $E' = E \bigcup \{(v,x) : v \in V, x \in V_v\}$. For each newly added edge (v,x), we set $w'(v,x) = B(v) \cdot (\beta - 1)/(k \cdot \beta)$, while for the edges $(u,v) \in E$, $w'(u,v) = w(u,v)$. For each $v \in V$, we set an initial battery capacity $B'(v) = \beta B(v)$. For each additional node $x \in V' \backslash V, B'(x) = 0$. Note that for each edge $(u,v) \in N', w'(u,v) \leq B'(u)\beta$. Therefore, the constructed instance $N' = (V', E', w', B')$ satisfies the desired condition.

We now show that the input instance $I = \langle N = (V, E, w, B),\ r \in V, k \geq 1\rangle$ of the decision version of the MTB problem is positive, if and only if, the constructed instance $I' = \langle N' = (V', E', w', B'), r, k \geq 1\rangle$ of the decision version of the restricted MTB problem is positive.

Suppose that the instance I of the decision version of the MTB problem is positive, i.e, it has a k-round MTB solution $\mathscr{T} = \{T_1, \ldots, T_k\}$. Let $T_i = (V, E_i)$. Consider broadcast trees $\mathscr{T}' = \{T_1', \ldots, T_k'\}$ in N' such that $T_i' = (V', E_i')$ and $E_i' = E_i \bigcup \{(v,x) : v \in V, x \in V_v\}$. It is easy to verify that these k broadcast trees satisfy the energy constraints in N', so I' is a positive instance of the decision version of the restricted MTB problem.

Conversely, assume that the instance I' of the decision version of the restricted MTB problem is positive, i.e, it has a k-round MTB solution $\mathscr{T}' = \{T_1', \ldots, T_k'\}$. Consider broadcast trees $\mathscr{T} = \{T_1, \ldots, T_k\}$ in N, where T_i is T_i' restricted to V. It is easy to verify that these k broadcast trees satisfy the energy constraints in N, so I is a positive instance of the decision version of the MTB problem.

Thus, it follows that the optimisation version of this problem is NP-hard. □

We now show in the following Theorem 6.7 that the restricted MTU problem is also NP-hard.

Theorem 6.7. *The* MTU *problem remains NP-hard even if the input instances are restricted by the condition that for each edge* (u,v), $w(u,v) \leq B(u)/\beta$, *where* $\beta > 1$ *is a fixed (but arbitrary) integer constant.*

Proof. Let MTU_β denote the decision version of the restricted MTU problem defined in the statement of the theorem. We define also the following different restriction of the MTU problem.

MTU-*3Part problem*

Instance: network $N = (V,E,w,B)$, and nodes $r,s \in V$. The network N has the following special structure, where c is a positive constant and the integer m indicates the size of network:

- $V = \{r,s,S_1,\ldots,S_m,v_1,\ldots,v_{3m}\}$,
- $E = E_1 \bigcup E_2 \bigcup E_3$,
  - $E_1 = \{(r,S_i) : i = 1,\ldots,m\}$,
  - $E_2 = \{(S_i,v_j) : i = 1,\ldots,m, j = 1,\ldots,3m\}$,
  - $E_3 = \{(v_j,s) : j = 1,\ldots,3m\}$,
- For all edges $(r,S_i) \in E_1$, $w(r,S_i) = 0$, and for all edges $(S_i,v_j) \in E_2$, $w(S_i,v_j) = x_j$, where $c/4 < x_j < c/2$, and $\sum_{j=1}^{3m} x_j = cm$. For all edges $(v_j,s) \in E_3$, $w(v_j,s) = c$,
- $B(r) = \infty, B(S_i) = c, i = 1,\ldots,m, B(v_j) = c, j = 1,\ldots,3m$.

Question: can the network N support $3m$ unicast rounds from the source r to destination s?

Bodlaender *et al.* [6] showed that this problem is strongly NP-Complete by a reduction from the 3-Partition problem. The sequence of numbers (edge-weights) $x_1,\ldots,x_{3m}$ and the

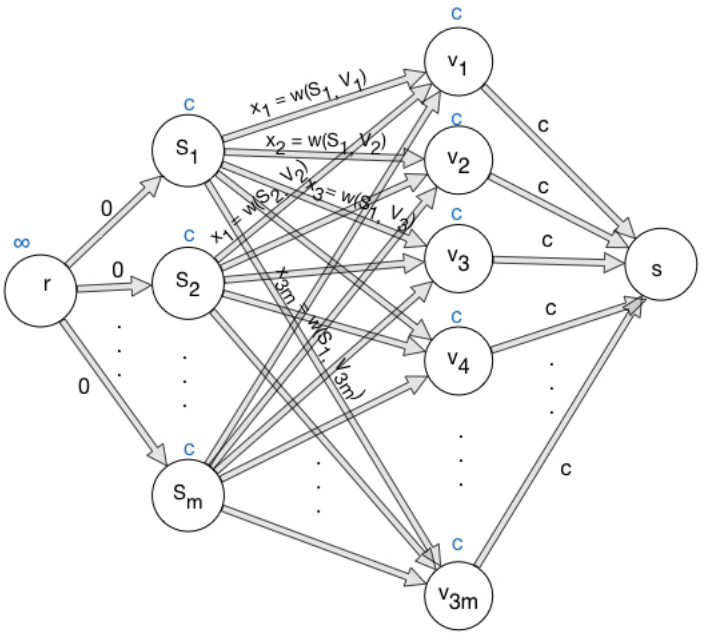


Figure 6.3 An instance of the MTU-3part problem, which is constructed from the instance of the 3-partition problem in pseudo-polynomial time.

number c are the input instance of the 3-Partition problem with the question whether $x_1, \ldots, x_{3m}$ can be partitioned into m subsets such that the sum of each subset is equal to c. Fig 6.3 shows the corresponding instance of the MTU-3Part problem. In the figure, the nodes $S_1, \ldots, S_m$ play the role of the subsets in the 3-partition problem.

We now reduce the MTU-3Part problem to MTU$_\beta$. The construction of an instance $I' = \langle N' = (V', E', w', B'), r, s, k' \rangle$ of MTU$_\beta$ from an instance $I = \langle N = (V, E, w, B), r, s \rangle$ of MTU-3Part is specified in Algorithm 1 and illustrated in Fig 6.4. Observe that for each edge $(u, v) \in E', w'(u, v) \leq B'(v)/\beta$. Therefore, I' is indeed an instance of MTU$_\beta$. Lemma 6.8 below implies that the constructed input instance I' of the decision version of the restricted MTU problem is also NP-complete. Therefore, it follows that the optimisation version of this problem is NP-hard. □

To complete the proof of Theorem 6.7, we prove the following lemma.

Lemma 6.8. *The instance I of* MTU*-3Part is positive, if and only if, the constructed instance I' of MTU$_\beta$ is positive.*

Input: an arbitrary instance $I = \langle N = (V, E, w, B), r, s \in V \rangle$ of MTU-3Part.
Output: instance $I' = \langle N' = (V', E', w', B'), r, s \in V', k' \rangle$, of MTU$_\beta$ (So, $w'(u,v) \leq B(u)/\beta$, for each $(u,v) \in E'$.
Construction of I':

1. The set of nodes V' is the union of the following sets:
   (a) $V = \{r,s\} \bigcup \{S_1, \ldots, S_m\} \bigcup \{v_1, \ldots, v_{3m}\}$ (the set of nodes in N).
   (b) $V_1 = \{p_1, \ldots, p_{3m}\}$.
   (c) $V_2 = \{t_1, \ldots, t_m\}$.
   (d) r is source and s is destination in N'
2. The set of edges E' is the union of the following sets.
   (a) E is the set of edges in N
   (b) $E_1 = \{(r, p_i) : i = 1, \ldots, 3m\}$. Source r has an outgoing edge to each node $p_i \in V_1$.
   (c) $E_2 = \{(S_i, t_i) : i = 1, \ldots, m\}$. For each node $S_i \in V$, there is an outgoing edge to t_i.
   (d) $E_3 = \{(p_i, v_i) : i = 1, \ldots, 3m\}$. For each node $p_i \in V_1$, there is an outgoing edge to v_i.
   (e) $E_4 = \{(t_i, s), i = 1, \ldots, m\}$. For each node $t_i \in V_2$, there is an outgoing edge to destination s.
3. We set edge-weights as follows.
   (a) For all edges $e \in E$, $w'(e) = w(e)$, that is as in N.
   (b) For all edges $e \in E_1$, $w'(e) = 0$
   (c) For all other edges e, $w'(e) = c$
4. We set battery capacity as follows.
   (a) For the source r, $B(r) = \infty$.
   (b) For all nodes $v \in V \setminus \{r,s\}$, we set $B(v) = 2\beta c$
   (c) For all nodes $v \in V_1 \cup V_2$, $B(v) = (2\beta - 1)c$.
5. $k' = 8m\beta - m$

Algorithm 1: Reduction from MTU-3Part to MTU$_\beta$

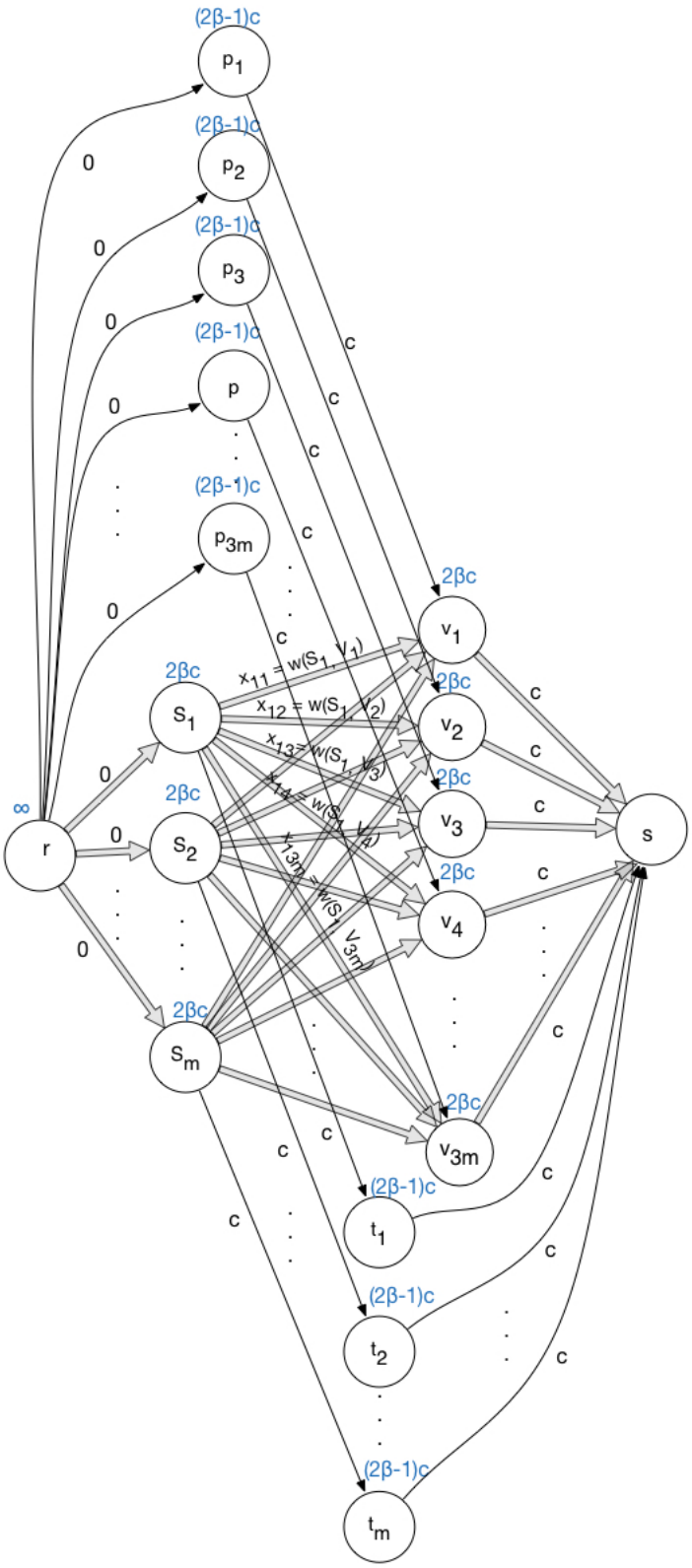


Figure 6.4 The instance of the restricted MTU problem

Proof. Assume that the instance $I = \langle N = (V,E,w,B), r, s\rangle$ of the MTU-3Part problem is positive, i.e, the network N supports $3m$ unicast rounds from source r to destination s. We show that the constructed instance $I' = \langle N' = (V',E',w',B'), r, s, k'\rangle$ of the MTU$_\beta$ problem is positive, i.e, N' supports $k' = 8m\beta - m$ rounds. Since network N supports $3m$ unicast rounds and N is a part of N', it is clear that N' can also support $3m$ rounds using the same paths as in N. After supporting these rounds, the remaining battery capacities of nodes $S_1, \ldots S_m, v_1, \ldots, v_{3m}$ are $2\beta c - c = (2\beta - 1)c$: each node v_j transmits exactly one message and the cost of each transmission is equal to c, so the battery capacity at v_j decreases by c. Thus each node v_j receives exactly one message from one of the nodes S_i, and the cost of transmitting this message is equal to x_j. This means that the total cost of transmitting $3m$ messages from the nodes S_i to the nodes v_j is equal to $\sum_{j=1}^{3m} x_j = mc$. Hence the battery capacity at each node S_i must also decrease by c.

By using the paths $\bigcup_{j=1}^{3m}\{r \to p_j \to v_j \to s\}$ and $\bigcup_{i=1}^{m}\{r \to S_i \to t_i \to s\}$, network N' can support further $(2\beta - 1)\cdot 3m + (2\beta - 1)\cdot m = 8m\beta - 4m$ rounds. Hence, in total network N' can support $3m + 8m\beta - 4m = 8m\beta - m$ rounds. Thus, I' is a positive instance of MTU$_\beta$.

Conversely, suppose that the constructed instance I' of the MTU$_\beta$ problem is positive, i.e, N' supports $k' = 8m\beta - m$ unicast rounds from source r to destination s. We consider the solution for I' (the $8m\beta$ unicast rounds), which maximises the total energy usage at nodes $p_1, \ldots p_{3m}$. We claim that this solution must use all energy at each node in $\{p_1, \ldots p_{3m}\}$. Assume to the contrary that there is a node p_j, which does not use all energy. There must be a round which uses a path $r \to S_i \to v_j \to s$, because node v_j must use all its energy. Replacing this path with $r \to p_j \to v_j \to s$, we are getting a solution with the same number of rounds but with more energy used at nodes $\{p_1, \ldots, p_{3m}\}$. This contradiction implies the claim. The claim implies that $(2\beta - 1)$ transmissions have to be made through each path $\{r \to p_j \to v_j \to s\}, j = 1, \ldots, 3m$, which gives $(2\beta - 1)\cdot 3m$ unicast rounds.

We now show that the instance I of the MTU-3Part problem is positive, i.e, the network N supports $3m$ unicast rounds from the source r to destination s. It is clear that $8m\beta - m$ transmissions must be made from the nodes $\{v_1, \ldots, v_{3m}\}\bigcup\{t_1, \ldots, t_m\}$ to destination s. Each

such transmission costs c and the total energy at the nodes in $\{v_1,\ldots,v_{3m}\}\bigcup\{t_1,\ldots,t_m\}$ is equal to $(8m\beta-m)c$, so all nodes in $\{v_1,\ldots,v_{3m}\}\bigcup\{t_1,\ldots,t_m\}$ must use all their energy. Since the initial energy at t_i is equal to $(2\beta-1)c$ and there is a unique r-s path passing through t_i $\{r\to S_i\to t_i\to s\}$, there must be exactly $(2\beta-1)$ unicast rounds which use this path.

Transmissions through the paths $\bigcup_{i=1}^{m}\{r\to S_i\to t_i\to s\}$ and $\bigcup_{j=1}^{3m}\{r\to p_j\to v_j\to s\}$ give $(2\beta-1)\cdot m+(2\beta-1)\cdot 3m=8m\beta-4m$ unicast rounds. Note that after supporting these rounds, the remaining battery capacities of nodes $S_1,\ldots,S_m,v_1,\ldots,v_{3m}$ are exactly c, but all other nodes in the network have empty batteries. Hence, what is left from network N' is exactly network N. Since we consider a solution in N' which support $8m\beta-m$ rounds, we conclude that $3m$ rounds must be performed through the network N. Thus, I is a positive instance of MTU-3Part. □

The complexity of the restricted MTC problem is given in the following theorem. Since this can be shown by reduction from the decision version of the restricted MTU problem (MTU$_\beta$) with the same reduction steps as in the proof of Theorem 6.5, the proof is omitted.

Theorem 6.9. *The* MTC *problem remains NP-hard even if the input instances are restricted by the condition that for each edge* (u,v), $w(u,v)\leq B(u)/\beta$, *where* $\beta>1$ *is a fixed (but arbitrary) integer constant.*

Chapter 7

The MNL broadcast, convergecast and unicast problems

In this chapter, we describe the approximation algorithms for the single topology broadcast (STB) and the multiple topology broadcast, convergecast, and unicast (MTB, MTC, and MTU) MNL problems. As discussed in Section 6.6, these four MNL problems are NP-hard, while the single topology convergecast and unicast (STC and STU) MNL problems are polynomial. The mixedcast MNL problems are discussed separately in Chapter 8. In Sections 7.1 – 7.3, we first consider the single and multiple topology broadcast (STB and MTB) and multiple topology convergecast (MTC) problems. In Section 7.4, we separately discuss the algorithm for the MTU problem.

Recall that the input to the MNL broadcast, convergecast, and unicast problems is a network $N = (V, E, w, B)$, and a node $r \in V$ for broadcast and convergecast or two nodes $r, s \in V$ for unicast. In the single topology version of the problems, the output is a routing topology R which supports k rounds of the specified communication task and satisfies the energy constraints (6.2). In the multiple topology version of the problems, the output is a collection of routing topologies $\mathscr{R} = \{R_1, ..., R_k\}$, which supports k rounds of the specified communication task and satisfies the energy constraints (6.1). The optimisation objective is to maximise the number of rounds k.

We solve the MNL broadcast and convergecast problems (STB, MTB, and MTC) by first finding a good value of k using binary search and LP-relaxation of the problems. Computed k is then used in the formulation of the corresponding DWDCN problem. For the single topology braodcast (STB), the approximate solution to the DWDCN problem, obtained using the algorithmic framework discussed in Section 3.4, gives an approximate solution (a tree) for this MNL problem. In the case of the multiple topology broadcast and convergecast problems (MTB and MTC) the solution to the DWDCN problem gives a subgraph, which satisfies the energy constraints (6.1). We obtain the desired collection of broadcast or convergecast trees by applying Edmonds' theorem for packing arborescences [12].

Let the k-MNL problem refer to the decision version of the MNL problem: *does a k-round solution for the given MNL problem exist?* We solve the MNL broadcast and convergecast problems by formulating the decision k-MNL problem as a DWDCN problem defined for a multigraph N_k (the multigraph N_k is constructed from original input network N by replacing each edge with its k copies). The value k is obtained by the initial binary search. Since this approach uses multigraphs, the running times of the multiple topology broadcast and convergecast algorithms turn out to be only pseudo-polynomial, because the value of k, and consequently the size of graph N_k can be exponential in the size of the input network N. Nutov *et al.* [54] mentioned that the multiple topology broadcast and convergecast algorithms can be run in "true polynomial" time by considering the "capacitated" version of the DWDCN algorithms, but, they did not provide details of this approach. For completeness, we present in Section 7.5 details of this capacitated approach.

7.1 DWDCN problems corresponding to MNL broadcast and convergecast problems

The DWDCN problems are in fact more general than needed in the context of the MNL broadcast and convergecast problems, so in this section, we introduce three special variants of

the DWDCN problems and review the DWDCN algorithms. The objective of the DWDCN problems is to find a minimum cost spanning subgraph H that satisfies specified connectivity requirements and weighted degree constraints (7.1). The input to these problems is a directed graph $G = (V, E, c, w)$, where V is a set of nodes, E is a set of edges, c is an edge-cost function, and w is an edge-weight function, and a subset $B \subseteq V$, degree bounds $b : B \to \mathbb{R}^+$ and a set function f on V that defines the desired connectivity requirements of the output subgraph. Recall that an (α, β)-approximation algorithm for DWDCN computes in polynomial-time a subgraph which satisfies the specified connectivity requirements but may violate the optimality of the cost by up to a factor of α and the degree bounds by up to a factor of β. The special cases of DWDCN problems considered in this chapter do not have edge costs and the objective is only to satisfy the weighted degree constraints as tightly as possible. Therefore we will have now only one approximation parameter β.

The three special variants of the DWDCN problems corresponding to the MNL problems are defined below.

Weighted Degree Constrained k-Outconnected (k-Inconnected) Subgraph, WDCKOS (WDCKIS)

Input: A directed weighted graph $G = (V, E, w)$, out-degree bounds $b : V \to \mathbb{R}^+$, a root $r \in V$, and a positive integer k.

Output: A k-edge-outconnected (k-edge-inconnected) spanning subgraph H of G with root r that satisfies the weighted out-degree constraints:

$$w(\delta_H^{out}(v)) \; = \sum \{w(e) : e \in \delta_H^{out}(v)\} \; \leq \; b(v), \text{ for each } v \in V. \tag{7.1}$$

Weighted Degree Constrained Out-Arborescence, WDCOA

Input: A directed weighted graph $G = (V, E, w)$, out-degree bounds $b : V \to \mathbb{R}^+$, and a root $r \in V$.

Output: An out-arborescence T of G rooted at r that satisfies the weighted out-degree constraints (7.1).

The WDCKOS and WDCKIS problems are used to solve the multiple topology broadcast (MTB) and convergecast (MTC) problems, respectively. The WDCOA problem is used to solve the single topology broadcast (STB) problem. We note that the WDCOA problem is actually the special case of the WDCKOS problem when $k = 1$.

The connectivity requirements of the WDCKOS problem is to find a k-edge-outconnected spanning subgraph H of G with root r. A graph H is k-edge-outconnected from r if there are k edge-disjoint paths from r to each node in H. It is well known that a graph H contains k edge-disjoint paths from node r to all other nodes in H, if and only if, $\delta^{in}_E(S) \geq k$ for every subset $\emptyset \neq S \subseteq V \backslash \{r\}$. Hence, the WDCKOS problem of finding k-edge-outconnected spanning subgraph H with root r that satisfies the weighted out-degree constraints (7.1) can be formulated as the feasibility problem of the following integer program $P^{OS}_{IP}(k, b : G)$ with variables $x(e), e \in E' = \{(u,v) \in E : w(u,v) \leq b(u)\}$:

$$
\begin{array}{rll}
x\left(\delta^{in}_{E'}(S)\right) \geq k & \text{for all} \quad \emptyset \neq S \subseteq V \backslash \{r\}, & (C) \\
\displaystyle\sum_{e \in \delta^{out}_{E'}(v)} x(e) w(e) \leq b(v) & \text{for all } v \in V, & (W) \\
x(e) \in \{0,1\}, & \text{for all } e \in E'. & (B)
\end{array}
$$

To make it clear that the integer program is constructed for graph G, we use the notation $P^{OS}_{IP}(k, b : G)$.

The DWDCN problem with intersecting supermodular function f includes as a special case the WDCKOS problem by setting $f(S) = k$ for every subset $\emptyset \neq S \subseteq V \backslash \{r\}$ and $f(S) = 0$, otherwise. We have shown in Section 3.2 (see Theorem 3.1) that there is a polynomial-time $(2,5)$-approximation algorithm for the DWDCN problem with intersecting supermodular function f and weighed out-degree constraints. Hence, it follows that there is a polynomial-time 5-approximation algorithm for the WDCKOS problem.

The connectivity requirements of the WDCKIS problem is to find, for a given G and a root $r \in V$, a k-edge-inconnected spanning subgraph H with root r. Analogously to the notion of the

k-edge-outconnected subgraph, a graph H is k-edge-inconnected with root r, if for each node $v \neq r$, there are k edge-disjoint paths from v to r. A graph H contains k edge-disjoint paths from v to r, if and only if, $\delta^{out}_H(S) \geq k$ for every subset $\emptyset \neq S \subseteq V \setminus \{r\}$. Hence, the WDCKIS problem of finding k-edge-inconnected spanning subgraph H with root r that satisfies the weighted out-degree constraints (7.1) can be formulated as the following integer program $P^{IS}_{IP}(k,b:G)$ with variables $x(e), e \in E' = \{(u,v) \in E : w(u,v) \leq b(u)\}$:

$$\begin{aligned} x(\delta^{out}_{E'}(S)) &\geq k, && \text{for all } \emptyset \neq S \subseteq V \setminus \{r\}, \\ \sum_{e \in \delta^{out}_{E'}(v)} x(e)w(e) &\leq b(v), && \text{for all } v \in V, \\ x(e) &\in \{0,1\}, && \text{for all } e \in E'. \end{aligned}$$

Let $G' = (V, E_r, w)$ be the reverse graph of $G = (V, E, w)$, where $E_r = \{(u,v) : (v,u) \in E\}$, and $w(u,v) = w(v,u)$ and $c(u,v) = c(v,u)$ for each edge $e \in E_r$. Since $|\delta^{out}_E(S)| = |\delta^{in}_{E_r}(S)|$ and $w(\delta^{out}_E(v)) = w(\delta^{in}_{E_r}(v))$, the WDCKIS problem on G is a special case of the DWDCN problem on G' with intersecting supermodular constraints such that $|\delta^{in}_E(S)| \geq k$ for all $\emptyset \neq S \subseteq V \setminus \{r\}$, and the weighted in-degree constraints. Theorem 3.2 for the DWDCN problem with intersecting supermodular function and weighted in-degree constraints implies that there is a polynomial-time 3-approximation algorithm for the WDCKIS problem.

The connectivity requirements of the WDCOA problem is to find an out-arborescence T of G. A graph G contains an out-arborescence rooted at r, if it contains a path from node r to every node in G. Hence, the WDCOA problem of finding an out-arborescence T of G rooted at r that satisfies the weighted degree constraints (7.1) can be formulated as the integer program $P^{OA}_{IP}(b:G) \equiv P^{OS}_{IP}(1,b:G)$. Thus the WDCOA problem is a special case of DWDCN problem with $0,1$-valued intersecting supermodular function f and weighted out-degree constraints. Nutov has shown in [52] that such DWDCN problem admits $(2,5)$-approximation algorithm, hence there is a polynomial-time 5-approximation algorithm for the WDCOA problem.

The algorithms for the WDCKOS, WDCKIS, and WDCOA problems are based on the framework described in Section 3.4. We first check if the LP-relaxation of the corresponding

integer programs, obtained by replacing binary constraints (B) with bounds $0 \leq x(e) \leq 1$, is feasible. Let $P_{LP}^{OS}(k,b:G)$, $P_{LP}^{IS}(k,b:G)$ and $P_{LP}^{OA}(b:G)$ denote the LP-relaxation of the corresponding IP problems. If there is no feasible solution for the LP-polytope considered, the computation terminates with output "INFEASIBLE", meaning that there is no subgraph that satisfies the desired connectivity requirements and the weighted degree constraints (7.1). Otherwise, we construct a subgraph H in the following incremental process. We maintain a set of edges J of subgraph H, which is initially empty. In each iteration, we consider a "residual" LP problem (formally defined in Section 7.5), compute a basic feasible solution for this problem, and on the basis of this solution, we remove some edges from the graph, or add some edges to set J, or remove some weighted degree constraints. The final set of edges J is the output of the computation: a subgraph of G which satisfies the desired connectivity requirements and violates the weighted degree constraints (7.1) by at most a factor of β. The value of β depends on the type of problem which we consider.

The following lemma follows from our results for the DWDCN problems given in Chapter 3 and Nutov's results given in [50, 52].

Lemma 7.1. *For each problem,* WDCKOS, WDCKIS, and WDCOA, *there exists a polynomial- time algorithm, which computes one of the following two outcomes.*

1. *Correctly determines that the LP-polytope corresponding to the input instance is empty.*

2. *If the LP-polytope is not empty, then the algorithm finds a k-edge-outconnected spanning subgraph H (*WDCKOS *problem) with root r, or k-edge-inconnected spanning subgraph H (*WDCKIS *problem) with root r, or an out-arborescence H (*WDCOA *problem), which violates the weighted degree constraints (7.1) by at most a factor of β, that is, for all $v \in V$,*

$$\sum\{w(e) : e \in \delta_H^{out}(v)\} \leq \beta \cdot b(v). \tag{7.2}$$

where:

- $\beta = 5$, *for the* WDCKOS *and* WDCOA *problem,*

- $\beta = 3$, *for the* WDCKIS *problem.*

The parts of this lemma which refer to problems WDCKOS, WDCKIS and WDCOA follow, respectively, from Theorems 3.1 and 3.2 and Nutov [50, 52]. Note that even if the input is infeasible (that is, the corresponding integer program is empty), algorithms of Lemma 7.1 may still output a subgraph, which satisfies the conditions given in part 2 of this lemma. In what follows, we refer to these polynomial-time algorithms for the WDCOA, WDCKOS, and WDCKIS problems as WDCOA, WDCKOS, and WDCKIS algorithms, respectively.

7.2 Decision versions of MNL broadcast and convergecast as DWDCN problems

In this section, we formulate the decision versions k-STB, k-MTB, and k-MTC of the STB, MTB, and MTC problems as the WDCOA, WDCKOS, and WDCKIS problems, respectively. The k-STB asks whether there is a k-round solution for the given STB problem. By setting the weighted out-degree constraints b as the energy constraints B, the k-STB problem can be formulated as integer program $P_{IP}^{OA}(B/k:N)$. The k-STB problem returns "YES", if and only if, integer program $P_{IP}^{OA}(B/k:N)$ has a feasible solution.

The following theorem is due to Edmonds [12]. Using this theorem, we obtain the solutions for the multiple topology broadcast and convergecast problems from the outputs of the WDCKOS and WDCKIS problems.

Theorem 7.2. *[12] Let $G = (V,E)$ be a directed graph with a specified root $r \in V$. The graph G contains k edge-disjoint spanning out-arborescences (in-arborescences) rooted at r, if and only if, G is k-edge-outconnected with r (k-edge-inconnected with r). Moreover, there is a polynomial-time algorithm that computes such k disjoint arborescences, if they exist.*

The fastest known algorithm for computing k edge-disjoint spanning out-arborescences from a k-edge-outconnected graph runs in $O(|E|k\log|V|+|V|k^4\log^2|V|)$ time [4]. The same bound applies also to k-edge-inconnected graphs and in-arborescences.

Edmonds' theorem says that a directed graph G contains k edge-disjoint out-arborescences rooted at r, if and only if, G is k-edge-outconnected with root r. Therefore, if we find a k-edge-outconnected (resp. k-edge-inconnected to r) spanning subgraph H of G with root r that satisfies the weighted out-degree constraints (7.1), then Edmonds' theorem implies that we can retrieve in polynomial-time k edge-disjoint out-arborescences (resp. in-arborescence) from H (hence, also from G).

In the view of the above, we can approach the k-MTB problem in the following way. Formulate the integer program $P_{IP}^{OS}(k,B:N_k)$, where $N_k=(V,E_k,w,B)$ is the multigraph obtained from original input network $N=(V,E,w,B)$ by replacing each edge with its k copies. The k-MTB problem is feasible, if and only if, integer program $P_{IP}^{OS}(k,B:N_k)$ is feasible. By Edmonds' theorem, we know that the solution for the integer program, which represents a k-edge-outconnected spanning subgraph of N_k rooted at r, contains k edge-disjoint (in N_k) out-arborescences which, together, satisfy the energy constraints (6.1). This implies that there are k out-arborescences in the input graph N (not necessarily edge-disjoint), which satisfy the energy constraints (6.1). An analogous approach applies to the k-MTC problem and the corresponding integer program $P_{IP}^{IS}(k,B:N_k)$.

7.3 Algorithms and their analysis for the MNL convergecast and broadcast problem

In this section, we provide the details of the approximation algorithms for the single topology broadcast (STB) and the multiple topology broadcast and convergecast (MTB and MTC) problems in details. We analyse these algorithms and prove the parts of Theorem 6.1, which refer to the STB, MTB, and MTC problems, that is, we prove the lower bound of $\lfloor k_{opt}/\beta \rfloor$ on the number of rounds returned by these algorithms.

7.3.1 Algorithm for the MNL convergecast and broadcast problem

The single topology broadcast (STB) algorithm consists of two computational phases: 1) finding the value of k using binary search, 2) applying the WDCOA algorithm of Lemma 7.1 to find a broadcast tree feasible for k rounds. In the MTB and MTC algorithms the second phase applies the WDCKOS and WDCKIS algorithms of Lemma 7.1, respectively. For these two algorithms we also have an additional phase of finding k broadcast or convergecast trees rooted at r from the output subgraphs of the WDCKOS and WDCKIS algorithms (Edmonds' theorem). We describe now each computational phase. In what follows, the constant β always refers to the corresponding approximation value stated in Theorem 6.1.

Finding the value of k

First consider the STB problem. It should be clear that k_{opt}^{STB} (the optimal k for STB problem) is the largest k such that integer program $P_{IP}^{OA}(B/k : N)$ is feasible ($k_{opt}^{STB} = 0$, if $P_{IP}^{OA}(B : N)$ not feasible). Let k^+ be the largest integer $k \geq 1$ such that LP-polytope $P_{LP}^{OA}(B/(k \cdot \beta) : N)$ is not empty or $k^+ = 0$, if $P_{LP}^{OA}(B/\beta : N)$ is empty. We find k^+ by binary search, checking in each iteration whether an LP-polytope $P_{LP}^{OA}(B/(k \cdot \beta) : N)$ is empty, using an appropriate polynomial-time LP algorithm.

Consider now the MTB (resp. MTC) problem and let k_{opt}^{MTB} (resp. k_{opt}^{MTC}) be the largest integer k such that the integer program $P_{IP}^{OS}(k, B : N_k)$ (resp. $P_{IP}^{IS}(k, B : N_k)$) is feasible. Let K be the largest integer $k \geq 1$ such that LP-polytope $P_{LP}^{OS}(k, B : N_k)$ is not empty, or $K = 0$ if $P_{LP}^{OS}(1, B : N)$ is empty. Similarly to the STB problem, we find K by binary search, checking in each iteration whether an LP-polytope $P_{LP}^{OS}(K, B : N_K)$ is empty. If $K \geq 1$, let k^* be the largest integer k such that the LP-polytope $P_{LP}^{OS}(k, B/\beta : N_K)$ is not empty (constants $\beta = 5$ for MTB and $\beta = 3$ for MTC, from Theorem 6.1). We again find the value of k^* by binary search, checking in each iteration whether an LP-polytope $P_{LP}^{OS}(k, B/\beta : N_K)$ is empty. It should be clear that $k^* \leq K$ for any $\beta > 1$.

It was shown in [54] that for all MNL problems $k_{opt} \leq k_{max}$, where,

$$k_{max} = \sum_{v \in V} \frac{B(v)}{\min\{w(e) : e \in \delta_E^{out}(v), w(e) > 0\}}. \tag{7.3}$$

The binary search for finding the values of k^+, k^*, and K can be done over the range $[0, k_{max}]$, that is in $O(\log k_{max})$ iterations, which is $O(\log(nL))$, where L is the largest integer in the representation of rational numbers $B(v)$ and $w(e)$ (observe that $k_{\max} \leq nL^2$). For this binary search, we are assuming that $w(e) > 0$ for at least one edge. Otherwise, trivially any number of rounds is feasible.

Applying the DWDCN algorithms of Lemma 7.1

First consider the STB problem. If $k^+ = 0$, then $k_{opt}^{STB} < \beta$ and the algorithm returns "0 rounds". Otherwise, if $k^+ \geq 1$, we apply the WDCOA algorithm of Lemma 7.1 with $b = B/(\beta \cdot k^+)$. Since the LP-polytope $P_{LP}^{OA}(B/(\beta \cdot k^+) : N)$ is not empty, Lemma 7.1 implies that the WDCOA algorithm returns a single out-arborescence. This out-arborescence and the integer k^+ (the number of rounds) are the output for the STB problem. Observe that this out-arborescence is feasible for the STB problem, because for $b = B/(\beta \cdot k^+)$ the conditions (7.2) are equivalent to the energy constraints (6.2):

$$k^+ \cdot \sum\{w(e) : e \in \delta_H^{out}(v)\} \leq B(v), \text{ for all } v \in V.$$

Now consider the MTB (resp. MTC) problem. If $K = 0$, then we set $k^* = 0$ and return an empty collection of trees. Otherwise, if $k^* \geq 1$, we apply the WDCKOS (resp. WDCKIS) algorithm of Lemma 7.1 to multigraph N_K with $b = B/\beta$. Since the LP-polytope $P_{LP}^{OS}(k^*, B/\beta : N_K)$ (resp. $P_{LP}^{IS}(k^*, B/\beta : N_K)$) is not empty by the definition of k^*, Lemma 7.1 implies that WDCKOS (resp. WDCKIS) algorithm returns a k^*-edge-outconnected (resp. k^*-edge-outconnected) spanning subgraph H of graph N_K. This k^*-edge-outconnected spanning subgraph with root r satisfies the energy constraints (6.1), because for $b = B/\beta$, the constraints

(7.2) are equivalent to:

$$\sum\{w(e) : e \in \delta_H^{out}(v)\} \leq B(v), \text{ for all } v \in V.$$

Finding disjoint trees for MTB and MTC problems

If $k^* \geq 1$, then the output of the WDCKOS (reps. WDCKIS) problem is a k^*-edge-outconnected (resp. k^*-edge-inconnected) spanning subgraph H of graph N_K with root r, which satisfies the energy constraints (6.1). From Edmonds' Theorem 7.2, we know that the output graph H of the WDCKOS (resp. WDCKIS) problem contains k^* edge-disjoint out-arborescences (resp. in-arborescences) rooted at r. We can retrieve such arborescences in time polynomial in the size of H. This gives us k^* out-arborescences (resp. in-arborescences) in graph N (not necessarily edge-disjoint), which satisfy the energy constraints (6.1). The solution to the MTB (resp. MTC) problem is the integer k^* and collection of k^* out-arborescences (resp. in-arborescences) rooted at r.

Running times of the STB, MTB and MTC algorithms

At the beginning of the STB algorithm, we find the value of k^+ using binary search, by checking in each iteration whether LP-polytope $P_{LP}^{OA}(B/(k \cdot \beta) : N)$ is empty or not. Since all these LP-polytopes have an exponential number of cut constraints (C), so to solve them in polynomial-time using the ellipsoid method, we need a polynomial-time separation oracle. The existence of such oracles in the general case of the DWDCN problems with intersecting supermodular connectivity requirements was discussed in Section 3.4. For the $P_{LP}^{OA}(B/\beta : N)$ polytopes, we can use the following specific separation oracle to determine a violated constraint.

Let x be the candidate point. For the n weighted out-degree constraints, we can check, one by one, each constraint if it is violated. This can be done in $O(n^2)$ time. Now consider the cut constraints. We set $x(e)$ as the capacity of edge e. If we have $x(\delta_E^{in}(S)) < 1$, for some $S \subseteq V\backslash\{r\}$, then $(V\backslash S, S)$ is an r-s cut of value less than 1 for any $s \in S$. Hence, the minimum r-s cut must have value less than 1. For every $s \in V\backslash\{r\}$, we consider the minimum

r-s cut problem. We can solve this problem by applying a polynomial-time maximum flow algorithm $n-1$ times, once for each node $s \in V\backslash\{r\}$. If for some $s \in V\backslash\{r\}$, the minimum r-s cut $(V\backslash S, S)$ has capacity less than 1, then we have found a violated cut inequality: the constraint corresponding to this set S is violated. This separation oracle can be implemented in $O(n^2) + O(n)M(m,n) = O(n)M(m,n)$ time, where $M(m,n)$ is time taken by the maximum flow algorithm. Thus, feasibility of the LP-polytopes $P_{LP}^{OA}(k, B : N)$ can be determined in polynomial-time using the ellipsoid method. This implies that the initial binary search for finding k^+ can be done in polynomial-time.

At each iteration of the WDCOA algorithm, we need to compute a basic feasible solution for the "residual" LP-polytope. We can do this by the ellipsoid method using the separation oracle as above. If we find a feasible solution, then we can convert it into a basic feasible solution in polynomial-time, following the algorithm given in [27]. Hence the WDCOA algorithm runs in polynomial-time and the running time of the overall STB algorithm is polynomial.

Now, consider the multiple topology broadcast and convergecast problems (MTB and MTC). For these problems, the corresponding LP-polytopes are constructed for multigraphs N_k, where k is the parameter of the binary search. In this setting, the LP-polytopes can be solved in pseudo-polynomial time because the values of k and consequently the size of graphs N_k can be exponential in the size of the input network N. Therefore, the running times of the overall MTB and MTC algorithms run in pseudo-polynomial time. In Section 7.5, we will show how this approach can be implemented to give polynomial running times.

7.3.2 Analysis of the algorithms for MNL broadcast and convergecast problems

It follows from the description of the algorithms in Section 7.3.1 that they return feasible solutions for the STB, MTB, and MTC problems. Now we prove Theorem 6.1 for the STB,

MTB, and MTC problems, that is, we prove the lower bound of $\lfloor k_{opt}/\beta \rfloor$ on the number of broadcast or convergecast trees returned by the algorithm (the MTU problem is separately considered in Section 7.4). As stated in Theorem 6.1 for the MTB and MTC problems, this lower bound holds only if the input instance N of the problem is such that $w(u,v) \leq B(u)/\beta$ for all edges $e \in N$. This condition is used in the lemma below because the LP-polytope $P_{LP}(k, B/\beta : G)$ is defined only for the edges in $E' = \{(u,v) \in E : w(u,v) \leq b(u)/\beta\}$.

Lemma 7.3. *Let $G = (V,E)$ be a directed graph and $k \geq 1$. If the LP-polytope $P_{LP}^{OS}(k, B : G)$ is not empty and $w(v,x) \leq B(v)$ for each edge $(v,x) \in E$, then the LP-polytope $P_{LP}^{OS}(\lfloor k/\beta \rfloor, B/\beta : G)$ is also not empty, for any $\beta \geq 1$.*

Proof. Let $\langle \bar{x}(e) \rangle_{e \in E}$ be a feasible solution for the LP-polytope $P_{LP}^{OS}(k, B : G)$. It is easy to show that $\langle \bar{\bar{x}}(e) \rangle_{e \in E} = \langle \bar{x}(e)/\beta \rangle_{e \in E}$ is a feasible solution for the LP-polytope $P_{LP}^{OS}(\lfloor k/\beta \rfloor, B/\beta : G)$, which is defined by the following constraints:

$$\begin{aligned} x(\delta_E^{in}(S)) &\geq \lfloor k/\beta \rfloor, && \text{for all } \emptyset \neq S \subseteq V \backslash \{r\}, && (C_2); \\ \sum_{e \in \delta_E^{out}(v)} x(e) w(e) &\leq B(v)/\beta, && \text{for all } v \in V, && (W_2); \\ 0 \leq x(e) &\leq 1, && \text{for all } e \in E. \end{aligned}$$

Indeed, since the vector $\langle \bar{x}(e) \rangle_{e \in E}$ satisfies the cut constraints (C_1), that is,

$$\bar{x}(\delta_E^{in}(S)) \geq k, \text{ for all } \emptyset \neq S \subseteq V \backslash \{r\},$$

we have,

$$\bar{\bar{x}}(\delta_E^{in}(S)) = \bar{x}(\delta_E^{in}(S))/\beta \geq \lfloor k/\beta \rfloor, \text{ for all } \emptyset \neq S \subseteq V \backslash \{r\},$$

so, $\langle \bar{\bar{x}}(e) \rangle_{e \in E}$ satisfies the cut constraints (C_2). Similarly it is easy to see that $\langle \bar{\bar{x}}(e) \rangle_{e \in E}$ satisfies the weighted degree constraints (W_2). Finally, because $\beta \geq 1$, we have,

$$0 \leq \bar{\bar{x}}(e) = \bar{x}(e)/\beta \leq 1/\beta \leq 1, \text{ for all } e \in E.$$

□

Analogous lemmas can be proven for the LP-polytopes $P_{LP}^{IS}(k, B : G)$ and $P_{LP}^{IS}(\lfloor k/\beta \rfloor, B/\beta : G)$. From now on, for simplicity of notation we drop the superscript of integer programs and their corresponding LP-polytopes, as these will be clear from the context. The following lemma applies to each of the two LP-polytopes P_{LP}^{OS} and P_{LP}^{IS}.

Lemma 7.4. *If the input instance N is such that edge-weight $w(u,v) \leq B(v)/\beta$ for all edges $e \in N$ and $K \geq 1$ in the* MTB *algorithm, then the LP-polytope $P_{LP}(\lfloor k_{opt}/\beta \rfloor, B/\beta : N_K)$ is not empty.*

Proof. Recall that k_{opt} is the largest integer k such that the integer program $P_{IP}(k, B : N_k)$ is feasible ($k_{opt} = 0$, if $P_{IP}(k, B : N)$ not feasible). By definition, K is the largest integer k such that $P_{LP}(k, B : N_k)$ is not empty. Hence, it is clear that $k_{opt} \leq K$. This implies that, if $K \geq 1$, the integer program $P_{IP}(k_{opt}, B : N_K)$ is also feasible. Thus, its LP-relaxation $P_{LP}(k_{opt}, B : N_K)$ is also feasible. Lemma 7.3 implies that the LP-polytope $P_{LP}(\lfloor k_{opt}/\beta \rfloor, B/\beta : N_K)$ is feasible. □

The lower bounds on the approximation factors for the multiple topology problems MTB, and MTC are given in the following two lemmas.

Lemma 7.5. *Let $\beta = 5$. The algorithm for the multiple topology broadcast problem* MTB *returns a solution with k^* broadcast trees such that $k^* \geq \lfloor k_{opt}/\beta \rfloor$, if the input instance N is such that edge-weight $w(u,v) \leq B(u)/\beta$ for all edges $e \in N$.*

Proof. We refer to the description of MTB algorithm. We show that the solution k^* is at least $\lfloor k_{opt}/\beta \rfloor$. The algorithm for the MTB problem always returns a feasible solution (either $k^* = 0$ or a collection of broadcast trees, which is feasible for k^* rounds). If $K = 0$, consequently $k^* = 0$, then $P_{IP}(1, B : N)$ is empty, so $k_{opt} = 0$. Now we consider $K \geq 1$. By definition, k^* is the largest integer $k \geq 0$ such that the LP-polytope $P_{LP}(k, B/\beta : N_K)$ is not empty. From Lemma 7.4, we know that the LP-polytope $P_{LP}(\lfloor k_{opt}/\beta \rfloor, B/\beta : N_K)$ is not empty. Thus, $\lfloor k_{opt}/\beta \rfloor \leq k^*$. □

Similarly to Lemma 7.5, one can prove the following lemma which gives the approximation factor for the MTC problem.

Lemma 7.6. *Let $\beta = 3$. The algorithm for the multiple topology convergecast (MTC) problem returns a solution with k^* convergecast trees such that $k^* \geq \lfloor k_{opt}/\beta \rfloor$, if the input instance N is such that edge-weight $w(u,v) \leq B(v)/\beta$ for all edges $e \in N$.*

The lower bound on the approximation factor for the single topology broadcast problem STB is given in the following lemma. Note that the proof of this lemma is not based on Lemmas 7.3 and 7.4, so we do not require the condition that the weights of the edges (v,x) are at most $B(v)/\beta$.

Lemma 7.7. *The algorithm for the single topology broadcast (*STB*) problem returns k^+ or $k^+ \geq 1$ and a broadcast tree which supports k^+ rounds. In both cases, $k^+ \geq \lfloor k_{opt}/\beta \rfloor$.*

Proof. We show that the solution k^+ is at least $\lfloor k_{opt}/\beta \rfloor$. The algorithm for the STB problem always returns a feasible solution (either $k^+ = 0$ or a broadcast tree, which is feasible for k^+ rounds), so $k^+ \leq k_{opt}$. If $k^+ = 0$, then $P_{LP}^{OA}(B/\beta : N)$ is empty, but if $k_{opt} \geq 1$, then $P_{IP}^{OA}(B/k_{opt} : N)$ is not empty. Therefore, if $k^+ = 0$, then $k_{opt} < \beta$, so in this case $0 = k^+ \geq \lfloor k_{opt}/\beta \rfloor = 0$. Assume now that $k^+ \geq 1$. The integer program $P_{IP}(B/k_{opt} : N)$ is feasible. This implies that the LP-polytope $P_{LP}(B/k_{opt} : N)$ is not empty as it is the LP-relaxation of $P_{IP}(B/k_{opt} : N)$. Recall the definition of k^+, which is the largest integer k such that the LP-polytope $P_{LP}(B/(\beta \cdot k) : N)$ is not empty. Therefore, we know that the LP-polytope $P_{LP}(B/(\beta \cdot (k^+ + 1)))$ is empty. Therefore, $k_{opt} < \beta \cdot (k^+ + 1)$, so $\lfloor k_{opt}/\beta \rfloor \leq k^+$. □

Lemmas 7.5 – 7.7 prove Theorem 6.1 for the STB, MTB, and MTC problems.

7.4 Algorithm for the MNL unicast problem

In this section, we provide an approximation algorithm for the multiple topology unicast (MTU) problem and prove Theorem 6.1 for the MTU problem, that is, we prove the lower bound of

$\lfloor k_{opt}/\beta \rfloor$ on the number of r-s paths returned by the algorithm. This will complete the proof of Theorem 6.1 (proofs for STB, MTB, and MTC are given in previous section).

Nutov [51] gives a polynomial-time algorithm for the MTU problem, which computes a solution with at least $\lfloor k^{MTU}_{opt}/\beta \rfloor \leq k^*$ rounds, where $\beta = 16$. This approach is based on the formulation of the MTU problem as the DWDCN problem with a special intersecting set function called a *ring function*. A set function h is a ring function if h is intersecting supermodular and there exists $s \in V$ such that for each $S \subseteq V \setminus \{s\}, h(S) = 0$. Following this approach, we can also show that the computed solution supports $k^* \geq \lfloor k^{MTU}_{opt}/3 \rfloor$ rounds, which is the same approximation bound as for the MTC problem. For the simplicity of exposition, we prove the approximation bound for the MTU problem through the polynomial-time reduction from the MTU problem to the MTC problem.

As shown in Section 6.6, the multiple topology unicast problem (MTU) can be reduced to the multiple topology convergecast problem (MTC). Thus, an approximation algorithm for the MTC problem can also be applied to solve the MTU problem. Let a network $N = (V, E, w, B)$ and two nodes $r, s \in V$ be the input to the MTU problem. Given this input instance $I = \langle N, r, s \rangle$, we construct an instance $I' = \langle N' = (V, E', w, B), r' \rangle$ of the MTC problem as described in Section 6.6. Now, we apply the MTC approximation algorithm presented in the previous section. This will return a collection of convergecast trees $\mathscr{T}_{in} = \{T_1, \ldots, T_{k^*}\}$, rooted at r' that satisfies the energy constraints (6.1). We then find an r-s path from each convergecast tree T_i. Since there are k^* convergecast trees we can get k^* paths. This collection of k^* r-s paths is the solution to the MTU problem.

We have shown in previous section that $\lfloor k^{MTC}_{opt}/\beta \rfloor \leq k^*$, where $\beta = 3$. The instance I' of the MTC problem has k rounds, if and only if, the instance I of the MTU problem has k rounds. Thus, $k^{\text{MTU}}_{opt} = k^{\text{MTC}}_{opt}$, implying that $\lfloor k^{MTU}_{opt}/\beta \rfloor \leq k^*$, where $\beta = 3$.

7.5 Polynomial time implementation for the multiple topology MNL broadcast and convergecast problems

In Sections 7.1 – 7.3, we provided pseudo-polynomial algorithms for the multiple topology broadcast and convergecast (MTB and MTC) problems. In this section, we provide details of the *capacitated approach*, which gives polynomial running times for these problems.

7.5.1 Capacitated version of WDCKOS and WDCKIS problems

We first define the meaning of k-edge-connectivity in the context of the capacitated versions of the WDCKOS and WDCKIS problems. Given a graph $G = (V, E, r)$, where V is a set of nodes, E is a set of edges, $r \in V$ is a distinguished node, and a positive integer k, we say that edge capacities $Y : E \to \mathbb{R}^+$ support *k-out-flows from r*, if and only if, for each node $v \in V \setminus \{r\}$, there is an integral flow of value k from node r to node v. Similarly, we say that edge-capacities Y support *k-in-flows to r*, if and only if, for each node $v \in V \setminus \{r\}$, there is an integral flow of value k from node v to node r. For example, for an integer $k \geq 1$ and a graph $G = (V, E, r)$, we considered the property that a subgraph (V, F) of G is k-edge-outconnected with r. Now, for an integer $k \geq 1$ and a graph $G = (V, E, r)$, we consider the property that edge capacities $Y : E \to \mathbb{R}^+$ support k-out-flows from r.

We note that the k-edge-out-connectivity becomes a special case of the notion of k-out-flows. A subgraph $H = (V, F)$ is k-edge-outconnected with r, if and only if, the edge capacities $Y_H : E \to \mathscr{R}^+$ such that $Y_H(e) = 1$ if $e \in H$ and $Y_H(e) = 0$, otherwise, support k-out-flows from r.

The capacitated version of the WDCKOS problem asks for finding integral edge capacities $Y = (y(e_1), \ldots y(e_m))$, which support k-out-flows from r and satisfy the following weighted "out-degree" constraints.

$$\sum \{y(e) \cdot w(e) : e \in \delta_E^{out}(v)\} \leq b(v), \text{ for all } v \in V. \tag{7.4}$$

Analogously, we define the capacitated version of the WDCKIS problem. We refer to these two problems as *Capacitated Weighted Degree Constrained k-Out-Flow* (CWDCKOF) and *Capacitated Weighted Degree Constrained k-in-Flow* (CWDCKIF) problems, respectively.

Below we give formal definitions of the CWDCKOF and CWDCKIF problems.

Capacitated Weighted Degree Constrained k-Out-Flow Problem (CWDCKOF)

Input: A directed weighted graph $G=(V,E,w)$, out-degree bounds $b:V\rightarrow\mathbb{R}^{+}$, a root $r\in V$, and a positive integer k.

Output: Integral edge-capacities $Y=(y(e_1),\ldots y(e_m))$, which support k-out-flows from r and satisfy the weighted out-degree constraints (7.4).

Capacitated Weighted Degree Constrained k-In-Flow Problem (CWDCKIF)

Input: A directed weighted graph $G=(V,E,w)$, out-degree bounds $b:V\rightarrow\mathbb{R}^{+}$, a root $r\in V$, and a positive integer k.

Output: Integral edge-capacities $Y=(y(e_1),\ldots y(e_m))$, which support k-in-flows to r and satisfy the weighted out-degree constraints (7.4).

The maximum-flow minimum-cut theorem implies that edge capacities $Y:E\rightarrow\mathbb{R}_{+}$ support k-out-flows from r, if and only if, the capacity of each cut $(V\backslash S,S)$ such that $S\subseteq V\backslash\{r\}$ is at least k. Therefore, the CWDCKOF problem of finding integral edge-capacities which support k-out-flows from r and satisfy the weighted degree constraints (7.4) can be formulated as the following integer program with variables $y(e)$ for $e\in E'=\{(u,v)\in E:w(u,v)\leq B(u)\}$.

$$
\begin{array}{rll}
y\left(\delta^{in}_{E'}(S)\right)\geq k, & \text{for all} \quad \emptyset\neq S\subseteq V\backslash\{r\}, & (F_{out})\\
\sum\limits_{e\in\delta^{out}_{E'}(v)} y(e)w(e)\leq b(v), & \text{for all } v\in V, & (W)\\
y(e)\in\{0,1,\ldots,k\}, & \text{for all } e\in E', & (EC).
\end{array}
$$

Following our previous notations scheme, we denote the above integer program by $P^{COF}_{IP}(k,b:G)$.

The CWDCKIF problem can be similarly formulated as an integer program. For this problem the cut constraints are replaced with

$$y\left(\delta^{out}_{E'}(S)\right) \geq k, \qquad \text{for all} \quad \emptyset \neq S \subseteq V\backslash\{r\}, \qquad (F_{in});$$

We refer to the resulting integer program as $P^{CIF}_{IP}(k,b:G)$.

7.5.2 Decision versions of MTB and MTC problems as CWDCKOF and CWDCKIF problems

In this section, we formulate the decision versions k-MTB and k-MTC of the MTB and MTC problems as the CWDCKOF and CWDCKIF problems, respectively.

The following theorem is the capacitated version of Edmonds' theorem (see Theorem 7.2) for packing arborescences.

Theorem 7.8. *[19] Let $G=(V,E)$ be a directed network with a specified root $r \in V$ and let Y be a non-negative integral edge capacity function. Graph G contains k out-arborescences (resp. in-arborescences) rooted at r, if and only if, Y supports k-out-flows from r (resp. k-in-flows to r). Moreover, there is a polynomial-time algorithm that computes such k arborescences in the form of at most $m-n-2$ distinct arborescences with their multiplicity.*

The k-MTB input instance $\langle N=(V,E,w,B)$ and $r \in V\rangle$ is feasible, if and only if, the integer program $P^{COF}_{IP}(k,B:N)$ is feasible. If $P^{COF}_{IP}(k,B:N)$ is feasible, then each solution is integral edge-capacities $Y=(y(e_1),\ldots y(e_m))$ which support k-out-flows from r and satisfy the weighted degree constraints (7.4). If we have such edge-capacities, then we can apply Theorem 7.8 to obtain solutions for the MTB and MTC problems. An analogous equivalence applies to the k-MTC problem and the integer program $P^{CIF}_{IP}(k,B:N)$.

7.5.3 Algorithms for the capacitated versions of WDCKOS and WDCKIS

In this section, we present the algorithms for the CWDCKOF and CWDCKIF problems. Our description refers only to the CWDCKOF problem, but the CWDCKIF problem has an analogous algorithm. We describe the algorithm in the context of the MTB problem, so compare the $P_{IP}^{OS}(k,B:N_k)$ formulation of this problem given in Section 7.2 with the $P_{IP}^{COF}(k,B:N)$ formulation given here.

The CWDCKOF problem has the same input as the WDCKOS problem, but since the problem has different objective, the output of the problem is somewhat different. A solution to the WDCKOS problem is defined by a function $x : E_k \rightarrow \{0,1\}$, where E_k is a set of edges in graph N_k, which specifies which subset of edges in E_k forms the output subgraph. On the other hand, a solution to the CWDCKOF problem is defined by an edge-capacity function $Y : E \rightarrow \mathbb{Z}^+$, where $\mathbb{Z}^+$ denotes the set of non-negative integers, which specifies the maximum amount of integral flow that can pass through an edge.

The WDCKOS approximation algorithm of Lemma 7.1 can be modified to an approximation algorithm for the CWDCKOF problem. We have not provided a detailed description of how the WDCKOS algorithm works, so we first describe this algorithm in detail and then explain how it can be modified for the CWDCKOF problem. The WDCKOS algorithm is essentially the DWDCN algorithm described in Section 3.4. We need to recall this algorithm now in the context of the WDCKOS problem to be able to explain how it can be used for the CWDCKOF problem.

The WDCKOS algorithm works as follows. It initially checks whether the LP-polytope $P_{LP}^{OS}(k,B:N_k)$ is empty. If it is empty, then the algorithm returns "INFEASIBLE" and terminates, meaning that there is no k-edge-outconnected subgraph H of N_k with root r which satisfies the weighted out-degree constraints (7.1). Otherwise, the algorithm performs the following iterative process whilst maintaining an edge-set J (initially $J = \emptyset$) and node-set W (initially $W = V$) of

out-degree bounds. In each iteration, compute first a basic feasible solution $x(e)_{e\in E_k}$ for the LP-polytope $P_{LP}^{OS}(k,B:N_k)$. Then remove from E_k all the edges with $x(e)=0$. Remove from E_k also all edges with $x(e)\geq 1/\alpha$, but add all these edges to set J ($\alpha=2$ for the WDCKOS problem and it is a fixed parameter). Now, we have only edges e with $0<x(e)<1/\alpha$ in the "residual" graph (V,E_k). The algorithm then removes node v from node-set W, if the degree of v is less than or equal to $\triangle$ ($\triangle=3$ for the WDCKOS problem and it is another fixed parameter). That is, the WDCKOS algorithm removes from W all nodes with $|\delta_{E_k}^{out}(v)|\leq\triangle$. Note that the nodes removed from W still remain in the graph.

The residual LP-polytope $P_{LP}^{OS}(k,B;J,W)$ is formulated for the updated sets E_k, W, and J and then solved in the next iteration. The residual LP-polytope $P_{LP}^{OS}(k,B;J,W)$ is defined by the following constraints:

$$\begin{aligned}
x\left(\delta_{E_k}^{in}(S)\right) &\geq k-|\delta_J^{in}(S)|, && \text{for all} \quad \emptyset\neq S\subseteq V\backslash\{r\},\\
\sum_{e\in\delta_{E_k}^{out}(v)} x(e)w(e) &\leq B(v)-w(\delta_J^{out}(v))/\alpha, && \text{for all} \quad v\in W,\\
0\leq x(e) &\leq 1, && \text{for all} \quad e\in E_k.
\end{aligned}$$

Observe that $P_{LP}^{OS}(k,B;\emptyset,W)$ is the LP-polytope $P_{LP}^{OS}(k,B:N_k)$ used in the first iteration.

This iterative process continues until there is no edge left in E_k. At the end of the computation, the algorithm outputs the final set J. Denoting this set by F, the subgraph $H=(V,F)$ is a k-edge-outconnected spanning subgraph of N_k with root r, which violates the weighted degree constraints (7.1) by at most a factor of β, where $\beta=\alpha+\triangle=5$. The pseudo-code of the WDCKOS approximation algorithm for the multigraph N_k is outlined in Figure 7.1.

Let $P_{LP}^{COF}(k,B:N)$ be the LP-relaxation of the $P_{IP}^{COF}(k,B:N)$, obtained by replacing edge-capacity constraints (EC) with $0\leq y(e)\leq k$. Given a vector Y and node-set W, let

Input: A network $N_k = (V, E_k, w, B)$, and a node $r \in V$
Output: A k-edge-outconnected spanning subgraph (V, F) of N_k with root r, which satisfies the weighted degree constraints (7.1)

1 Initialization: $J \leftarrow \emptyset$, $W \leftarrow V$, $E_k \leftarrow E_k \backslash \{(v,u) \in E_k : w(v,u) > B(v)\}$,
2 **if** $P(k, B : N_k) = \emptyset$ **then**
3 | Return "INFEASIBLE" and terminates.
4 **while** $E_k \neq \emptyset$ **do**
5 | Find a basic feasible solution $x = (x(e))_{e \in E_k} \in P(k, B; J, W)$.
6 | Remove from E_k all edge with $x(e) = 0$.
7 | Add to J and remove from E_k all edges with $x(e) \geq 1/2$.
8 | Remove from W every $v \in W$ with $|\delta_E^{out}(v)| \leq 3$
9 **end**
10 $F \leftarrow J$;

Figure 7.1 The WDCKOS algorithm of Lemma 7.1 for the multigraph N_k

$P_{LP}^{COF}(k, B; Y, W)$ be defined by the following constraints.

$$y\left(\delta_E^{in}(S)\right) \geq k - \sum_{e \in \delta^{in}(S)} Y(e), \quad \text{for all} \quad \emptyset \neq S \subsetneq V \backslash \{r\}, \qquad (F_{out})$$

$$\sum_{e \in \delta_E^{out}(v)} y(e) w(e) \leq B(v) - \sum_{e \in \delta^{out}(v)} Y(e) \cdot w(e)/\alpha, \quad \text{for all } v \in V, \qquad (W)$$

$$0 \leq y(e) \leq 1, \quad \text{for all } e \in E, \qquad (EC).$$

We modify the WDCKOS algorithm as follows. Let us refer to the modified algorithm as the CWDCKOF algorithm. The CWDCKOF algorithm has similar procedures as the WDCKOS algorithm of Figure 7.1, but instead of solving the LP-polytope $P_{LP}^{OS}(k, B : N_k)$ in the first iteration, the CWDCKOF algorithm solves $P_{LP}^{COF}(k, B; N)$. In each subsequent iteration, the CWDCKOF algorithm now solves the residual LP-polytope $P_{LP}^{COF}(k, B; Y, W)$ instead of $P_{LP}^{OS}(k, B; J, W)$. The pseudo-code of the CWDCKOF algorithm is given in Figure 7.2. We present the pseudo-code by separating it into two parts: (i) 1st iteration (lines 7 – 17) and (ii) the 2nd and subsequent iterations (lines 19 – 24).

Although we presented the CWDCKOF algorithm in Figure 7.2, in order to show that the CWDCKOF algorithm has the same approximation property as the WDCKOS algorithm,

Input: A network $N = (V, E, w, B)$, and a node $r \in V$
Output: Integral capacities $Y = Y(e_1), \ldots, Y(e_m)$
1 Initialization: $Y(e) \leftarrow 0$ for each $e \in E$, $W \leftarrow V$, $E \leftarrow E \backslash \{(vu) \in E : w(vu) > b(v)\}$,
2 $\alpha = 2$ and $\triangle = 3$.
3 **if** $P_{LP}^{COF}(k, B : N) = \emptyset$ **then**
4 Return "INFEASIBLE" and STOP.
5 **else**
6 /* 1st iteration */
7 Find basic feasible solution $y \in P_{LP}^{COF}(k, B : N)$.
8 **forall the** $e \in E$ **do**
9 $Y(e) \leftarrow \lfloor y(e) \rfloor$;
10 **end**
11 Remove from E all edges with $y(e)$ integral;
12 **foreach** $e \in E$ **do**
13 **if** $y(e) - Y(e) \geq 1/\alpha$ **then**
14 $Y(e) = Y(e) + 1$;
15 Remove e from E;
16 **end**
17 Remove from W every $v \in W$ with $\delta_E^{out}(v) \leq \triangle$.
18 /* 2nd and subsequent iterations */
19 **while** $E \neq \emptyset$ **do**
20 Find basic feasible solution $y \in P_{LP}^{COF}(k, B; Y, W)$.
21 Remove from E all edge with $y(e) = 0$.
22 Set $Y(e) \leftarrow Y(e) + 1$ and remove e from E for each edge e with $y(e) \geq 1/\alpha$.
23 Remove from W every $v \in W$ with $|\delta_E^{out}(v)| \leq \triangle$
24 **end**

Figure 7.2 The CWDCKOF algorithm obtained from the WDCKOS algorithm

we obtain the solutions of the CWDCKOF algorithm using the WDCKOS algorithm in the following way.

First, we execute the first iteration of the CWDCKOF algorithm given in Figure 7.2 (execute lines 3 – 17 of the CWDCKOF algorithm). Let $y = (y(e))_{e \in E}$ be the computed basic feasible solution of the LP-polytope $P_{LP}^{COF}(k, B : N)$ obtained during the first iteration. From y we construct a basic feasible solution $x = (x(e_1^1), \ldots x(e_1^k), \ldots, x(e_m^1), \ldots x(e_m^k))$ for the LP-polytope $P_{LP}^{OS}(k, B : N_k)$ as follows.

For each edge $e \in E$ in N, let $e^1,\ldots,e^k$ be the k copies of e in multigraph N_k. For each edge e, we set $x(e^1),\ldots,x(e^k)$ in the following way. Let p be the integer part of $y(e)$ and q be the fractional part of $y(e)$, so $p+q=y(e)$. We set $x(e^i)=1$, for each i up to p and $x(e^{p+1})=q$. We then set $x(e)=0$ for all the remaining edges $e^{p+2},\ldots,e^k$. Note that now $x=(x(e_1^1),\ldots x(e_1^k),\ldots,x(e_m^1),\ldots x(e_m^k))$ satisfies the cut constraints (C), weighted degree constraints (W), and binary constraints (B) of the LP-polytope $P_{LP}^{OS}(k,B:N_k)$. Therefore, x is a feasible solution. Note that x is also a basic feasible solution since $x(e^1),\ldots,x(e^k)$ has at most one e^i, $0<x(e)<1$ and all other values $x(e^j)$, $j\neq i$, are either 0 or 1. If x does not satisfy this property then x is a linear combination of two distinct feasible solutions, so x is not a basic feasible solution.

From the constructed basic feasible solution x, we execute lines (4 – 9) of the WDCKOS algorithm, that is, in subsequent iterations of the CWDCKOF algorithm, we solve the residual LP-polytope $P_{LP}^{OS}(k,B;J,W)$, but instead of executing line 7, we execute line 22 of Figure 7.2, that is, we set $Y(e)\leftarrow Y(e)+1$ if its corresponding edge e^i has $x(e^i)\geq 1/2$ and remove from $e\in E$ if its corresponding edge e^i has $x(e^i)\geq 1/2$. We repeat this process until E is empty.

We claim that using this approach we can derive the solutions of the CWDCKOF algorithm. Note that at the end of the first iteration in both algorithms (CWDCKOF and WDCKOS), the constraints of the residual LP-polytopes $P_{LP}^{COF}(k,B:Y,W)$ are equivalent to the constraints of the $P_{LP}^{OS}(k,B:J,W)$, that is,

$$k-\sum_{e\in\delta^{in}(S)} Y(e) \;=\; k-|\delta_J^{in}(S)|, \qquad \text{for all} \quad \emptyset\neq S\subset V\backslash\{r\},$$

$$B(v)-\sum_{e\in\delta^{out}(v)} Y(e)\cdot w(e)/\alpha \;=\; B(v)-w(\delta_J^{out}(v))/\alpha, \qquad \text{for all } v\in V.$$

At most one copy e^i of edge e is left in the residual graph of N_k in the WDCKOS algorithm (the copy for which $0<x(e^i)<1/\alpha$). If there is one copy e^i of edge e left in the residual graph of N_k in the WDCKOS algorithm, then edge e is also left in the residual graph of N in the CWDCKOF algorithm. Hence the same set of edges are in both residual graphs. For example, if $y=(2.1,0,1.25,1.7,0.2)$ for edges $e_1,\ldots,e_5$, then at the end of the first iteration

of the CWDCKOF algorithm, $Y = (2,0,1,2,0)$, the remaining edges in the residual graph of N are e_1, e_3, e_5. Then, the constructed basic feasible solution x from y is

i	e_1^i	e_2^i	e_3^i	e_4^i	e_5^i
1	1	0	1	1	0.2
2	1	0	0.25	0.7	0
3	0.1	0	0	0	0

Therefore, at the end of the first iteration of the WDCKOS algorithm, $J = \{e_1^1, e_1^2, e_3^1, e_4^1, e_4^2\}$ and the remaining edges in the residual graph of N_k are e_1^3, e_3^2, e_5^1. Observe that J contains exactly $Y(e)$ copies of edge e for each $e \in E$ and the same set of edges are left in the residual graphs. Thus, we can consider $P_{LP}^{OS}(k, B : J, W)$ in subsequent iterations, instead of $P_{LP}^{COF}(k, B : Y, W)$. This implies that the CWDCKOF algorithm has the same approximation property as the WDCKOS algorithm.

Similarly to the STB problem, we can design a polynomial-time separation oracle for the CWDCKOF algorithm of Figure 7.2 and WDCKOS algorithm of Figure 7.1. If we find a feasible solution using the ellipsoid method with the separation oracle, we can convert it into a basic feasible solution by using the algorithm given in [27]. Hence, the running times of these algorithms are polynomial in the size of the graph. Thus, we have the following lemma.

Lemma 7.9. *For the* CWDCKOF *(resp.* CWDCKIF*) problem, there exists a polynomial-time algorithm, which computes one of the following two outcomes.*

1. *Correctly determines that the LP-polytope corresponding to the input instance is empty.*
2. *If the LP-polytope is not empty, then the algorithm finds integral edge-capacities Y which support k-out-flows (resp. k-in-flows) from r and violate the weighted degree constraints (7.4) by at most a factor of* $\beta = 5$ *(resp.* $\beta = 3$*)*

7.5.4 Algorithms for MNL broadcast and convergecast problems

We now show how the CWDCKOF and CWDCKIF algorithms can be used to find the approximate solutions for the MTB and MTC problems. We consider in detail only the MTB problem. A similar approach applies also to the MTC problem.

The MTB algorithm based on the capacitated approach has the same procedure as the "uncapacitated approach" (given in Section 7.3.1). Let k^* be the largest integer k such that $P_{LP}^{COF}(k, B/\beta : N)$ is not empty ($k^* = 0$ if $P_{LP}^{COF}(1, B/\beta : N)$ is empty). Similarly to the uncapacitated approach, we find k^* by binary search, checking in each iteration whether a polytope $P_{LP}^{COF}(k, B/\beta : N)$ is empty or not. If $k^* = 0$, we return an empty collection of trees. Otherwise, if $k^* \geq 1$, we apply the CWDCKOF algorithm of Lemma 7.9 to input graph N with $b = B/\beta$. Since the LP-polytope $P_{LP}^{COF}(k^*, B/\beta : N)$ is not empty by the definition of k^*, we get integral capacities Y which support k^*-out-flows from r, which satisfy the energy constraints (6.1). We apply the capacitated version of Edmonds' algorithm (Theorem 7.8) to obtain k^* out-arborescences. The output of the capacitated version of Edmonds' algorithm is a sequence of distinct out-arborescences $T_1, \ldots, T_p$ rooted at r and their multiplicities $\mathscr{M}(T_1), \ldots, \mathscr{M}(T_p)$, where $p \leq m - n - 2$ and $\sum_1^p \mathscr{M}(T_i) = k^*$. This (compact) representation of distinct out-arborescences is the solution for the MTB problem.

The initial binary search which needs to check the feasibility of polytopes $P_{LP}^{COF}(k, B/\beta : N)$ can be done in polynomial-time using the ellipsoid method. As stated in Section 7.5.2 (Theorem 7.8), there is a polynomial-time algorithm for packing arborescences [19]. Thus, combining all these computations, the running time of the overall MTB algorithm is polynomial in the size of the input graph N.

Chapter 8

The MNL mixedcast problems

In this chapter, we consider the mixedcast MNL problems. Nutov and Segal [54] introduced this problem and proposed an approximation algorithm based on the approximation algorithms for the MNL broadcast and convergecast problems, presented in the previous chapter. We first follow the approach in [54] and give a simple method, which yields an approximation factor of $\beta_B + \beta_C$, where β_B and β_C are the approximation factors of algorithms for broadcast and convergecast problems, respectively (see Table 6.1). We then introduce a new approach, which improves the approximation factor to $\max\{\beta_B, \beta_C\}$.

8.1 A Simple Method

Nutov and Segal [54] proposed solving the mixedcast problem by splitting the battery capacity at each node into two parts in proportion $\beta_B : \beta_C$. One part is used for the computation of broadcast trees and the other for convergecast trees. The approximation factor of the resulting algorithm is $\beta_B + \beta_C$. This approach, together with the approximation factors β_B and β_C gives the following approximation bound.

Lemma 8.1. *For the* MNL *mixedcast problems, we can find in polynomial time solutions with values* $k \geq \lfloor k_{opt}/\beta_M \rfloor$*, where* $\beta_M = \beta_B + \beta_C$*, for all input instance* N *such that edge-weight* $w(u,v) \leq B(u)/\beta_M$ *for each edge* (u,v)*. So,* $\beta_M = 6$ *for the single topology mixedcast (*STB*) problem and* $\beta_M = 8$ *for the multiple topology mixedcast (*MTM*) problem.*

Proof. We give a proof only for the multiple topology mixedcast (MTM) problem. The single topology mixedcast (STM) problem can be proven in a similar way.

Let β_B and β_C denote the values of β for the multiple topology broadcast and convergecast problems, that is, $\beta_B = 5$ and $\beta_C = 3$. Let B_M denote the node battery capacity function of the MTM problem and let k^* denote our solution for the mixedcast problem. The algorithm works as follows. We apply the MTB algorithm of Theorem 6.1 to the root node r_b and the battery capacity function

$$B_B = \frac{\beta_B}{\beta_B + \beta_C} B_M.$$

As a result, we compute a collection of broadcast trees $\mathscr{T}_{\mathscr{B}} = \{T_1, T_2, ..., T_{k_B^*}\}$ rooted at node r_b, which use at most $B_B(v)$ energy at each node v. Similarly, we apply the MTC algorithm of Theorem 6.1 to the root node r_c and the battery capacity function

$$B_C = \frac{\beta_C}{\beta_B + \beta_C} B_M.$$

We get a collection of convergecast trees $\mathscr{T}_{\mathscr{C}} = \{T_1, T_2, ..., T_{k_C^*}\}$ rooted at node r_c, which use at most $B_C(v)$ energy at each node v. Hence, the total energy used by the k_B^* broadcast trees and k_C^* convergecast trees is bounded by the initial battery capacity B_M, i.e, all these trees together satisfy the energy constraints (6.3). Our solution to the mixedcast problem MTM is

$$k^* = \min\left\{ \left\lfloor \frac{k_B^*}{\tau} \right\rfloor, \left\lfloor \frac{k_C^*}{\gamma} \right\rfloor \right\},$$

and the first $T_1, T_2, ..., T_{\tau k^*}$ broadcast trees from $\mathscr{T}_{\mathscr{B}}$ and the first $T_1, T_2, ..., T_{\gamma k^*}$ convergecast trees from $\mathscr{T}_{\mathscr{C}}$.

Now we show that $k^* \geq \lfloor k_{opt}/\beta_M \rfloor$, where k_{opt} is the optimal value of k for the multiple topology mixedcast (MTM) problem. We note that k is feasible for the MTB and MTC problem, if and only if, the integer programs $P_{IP}^{OS}(k, B : N_k)$ and $P_{IP}^{IS}(k, B : N_k)$ are not empty, respectively. For easy understanding of notations, in the rest of this chapter 8 we denote these integer programs by $P_{IP}^{MTB}(k, B)$ and $P_{IP}^{MTC}(k, B)$, respectively, and their corresponding LP-polytopes by $P_{LP}^{MTB}(k, B)$ and $P_{LP}^{MTC}(k, B)$.

The algorithm for the multiple topology mixedcast (MTM) problem always returns a feasible solution: either $k^* = 0$, or $k^* \geq 1$, and a collection of broadcast trees and convergecast trees that are feasible for k^* rounds. This implies that if $k_{opt} = 0$, $k^* = 0$. Assume now that $k_{opt} \geq 1$. This implies that τk_{opt} broadcast and γk_{opt} convergecast rounds can be performed within the battery capacity function B_M. This implies that the integer programs $P_{IP}^{MTC}(\gamma k_{opt}, B_M)$ and $P_{IP}^{MTB}(\tau k_{opt}, B_M)$ are not empty.

Lemma 7.3 implies that the following polytope

$$P_{LP}^{MTB}\left(\left\lfloor \frac{\tau \cdot k_{opt}}{\beta_B + \beta_C} \right\rfloor, \frac{B_M}{\beta_B + \beta_C}\right)$$

is not empty. Hence,

$$k_B^* \geq \left\lfloor \frac{\tau \cdot k_{opt}}{\beta_B + \beta_C} \right\rfloor \geq \tau \left\lfloor \frac{k_{opt}}{\beta_B + \beta_C} \right\rfloor.$$

Similarly, the following polytope

$$P_{LP}^{MTC}\left(\left\lfloor \frac{\gamma \cdot k_{opt}}{\beta_B + \beta_C} \right\rfloor, \frac{B_M}{\beta_B + \beta_C}\right),$$

is not empty. This implies that

$$k_C^* \geq \left\lfloor \frac{\gamma \cdot k_{opt}}{\beta_B + \beta_C} \right\rfloor \geq \gamma \left\lfloor \frac{k_{opt}}{\beta_B + \beta_C} \right\rfloor.$$

Hence,

$$k^* = \min\left\{ \left\lfloor \frac{k_C^*}{\gamma} \right\rfloor, \left\lfloor \frac{k_B^*}{\tau} \right\rfloor \right\} \geq \left\lfloor \frac{k_{opt}}{\beta_B + \beta_C} \right\rfloor = \left\lfloor \frac{k_{opt}}{\beta_M} \right\rfloor.$$

□

8.2 An Improved Method

Now we show a new approach for the mixedcast problems, which gives Theorem 6.4. In the previous approach, the battery capacities B are split into two fixed parts: $\frac{\beta_b}{\beta_b+\beta_c}$ fraction for broadcast and the remaining portion for convergecast. This partition is the same at each node. Our approach now is to replace this fixed a priori partition with a computed, more efficient partition, which does not have to be the same at all nodes. We discuss here only the multiple topology problem MTM; algorithm and its analysis for the STM problem are analogous.

Let $\beta_M = \max\{\beta_B, \beta_C\}$. Let $P_{IP}^{MTM}(k,B)$ be the following integer program, with variables $x(e)$ and $y(e), e \in E' = \{(u,v) \in E_k : w(u,v) \leq B(v)\}$ and $\bar{k} = \gamma k + \tau k$:

$$
\begin{array}{rcll}
x(\delta_E^{out}(S)) & \geq & \gamma k, & \text{for all } \emptyset \neq S \subseteq V \backslash \{r\}, \\
y(\delta_E^{in}(S)) & \geq & \tau k, & \text{for all } \emptyset \neq S \subseteq V \backslash \{r\}, \\
\sum_{e \in \delta_E^{out}(v)} x(e)w(e) + \sum_{e \in \delta_E^{out}(v)} y(e)w(e) & \leq & B(v), & \text{for all } v \in V, \\
x(e), y(e) & \in & \{0, \ldots, k\}, & \text{for all } e \in E.
\end{array}
$$

It is clear that there is a k-round feasible solution for MTM, if and only if, above integer program is feasible. We denote the LP-relaxation of the above IP by $P_{LP}^{MTM}(k,B)$. Let k' be the largest integer k such that the polytope $P_{LP}^{MTM}(k, B/\beta_M)$ is not empty.

The algorithm for MTM first finds k' using binary search, solving in each iteration a linear relaxation of the current integer program. If $k' = 0$, we output the empty collections of convergecast and broadcast trees. If $k' \geq 1$, then the algorithm also finds a feasible solution $\langle \bar{x}(e) \rangle_{e \in E}$ and $\langle \bar{y}(e) \rangle_{e \in E}$ for the polytope $P_{LP}^{MTM}(k', B/\beta_M)$. Let $B_{\bar{x}(v)}$ (resp., $B_{\bar{y}(v)}$) be the

fraction of the battery capacity $B(v)/\beta_M$ which is used by the broadcast part $\langle \overline{x}(e)\rangle_{e\in E}$ of the solution (resp., by the convergecast part $\langle \overline{y}(e)\rangle_{e\in E}$ of the solution). That is,

$$B_{\overline{x}}(v) = \sum_{e\in\delta_E^{out}(v)} \overline{x}(e)w(e), \quad B_{\overline{y}}(v) = \sum_{e\in\delta_E^{out}(v)} \overline{y}(e)w(e).$$

Now we apply the approximation algorithm for the CWDCKOF problem given in Lemma 7.9 to the (non-empty) set $P_{IP}^{MTB}(\tau k', B_{\overline{x}})$ and we apply the approximation algorithm for the CWDCKIF problem to the (non-empty) set $P_{IP}^{MTC}(\gamma k', B_{\overline{y}})$. This way we compute a collection of broadcast trees $\mathscr{T}_{\mathscr{B}} = \{T'_1, .., T'_{\tau k'}\}$ rooted at node r_b, which use at most $\beta_B B_{\overline{x}}(v)$ energy at each node v, and a collection of convergecast trees $\mathscr{T}_{\mathscr{C}} = \{T''_1, .., T''_{\gamma k'}\}$ rooted at node r_c, which use at most $\beta_C B_{\overline{y}}(v)$ energy at each node v. The output of our MTM algorithm is $(k', \mathscr{T}_{\mathscr{B}}, \mathscr{T}_{\mathscr{C}})$. This output is feasible, because the energy usage at each node v is at most

$$\begin{aligned}\beta_B B_{\overline{x}}(v) \;+\; \beta_C B_{\overline{y}}(v) &\leq\; \beta_M (B_{\overline{x}}(v) \;+\; B_{\overline{y}}(v)) \\ &\leq \beta_M (B(v)/\beta_M) \;=\; B(v).\end{aligned}$$

We now show that

$$k' \geq \lfloor k_{opt}/\beta_M \rfloor. \tag{8.1}$$

If $k_{opt} = 0$, then $P_{LP}^{MTM}(k, B/\beta_M)$ is empty for each $k \geq 1$, so $k' = 0$ and (8.1) holds. Assume now that $k_{opt} \geq 1$. The set $P_{IP}^{MTM}(k_{opt}, B)$ is not empty, so the LP polytope $P_{LP}^{MTM}(k_{opt}, B)$ is not empty. Lemma 8.2 (below) implies that the LP polytope $P_{LP}^{MTM}(\lfloor k_{opt}/\beta_M \rfloor, B/\beta_M)$ is not empty, and k' has been computed as the largest integer k such that $P_{LP}^{MTM}(k, B/\beta_M)$ is not empty, so (8.1) holds also in this case.

This concludes the proof of Theorem 6.4 (the MTM part). We used the following lemma, which is analogous to Lemma 7.3, so we omit its proof.

Lemma 8.2. *If the polytope $P_{LP}^{MTM}(k, B)$ is not empty, then the polytope $P_{LP}^{MTM}(\lfloor k/\beta \rfloor, B/\beta)$ is also not empty, for any $\beta \geq 1$. The analogous property holds for the P_{LP}^{STM} polytopes.*

It should be clear that our approach can be extended to more general mixedcast problems mentioned in Chapter 1.2. If, for example, we require that convergecast, broadcast and unicast tasks are periodically performed in proportion $\gamma : \tau : \eta$, then we extend the integer program $P_{IP}^{MTM}(k,B)$ by adding variables $z(e)$, $e \in E$, and the cut constraints appropriate for the unicast problem. The approximation factor of the obtained algorithm is equal to $\max\{\beta_B, \beta_C, \beta_U\}$, which currently is equal to 5 (see Table 6.1). The mixedcast can also contain a number of different communication tasks of the same type. Since in our model we may have at most n different broadcast tasks, n different convergecast tasks and n^2 different unicast tasks, then also in this general case we have a polynomial time algorithm with the same approximation factor $\max\{\beta_B, \beta_C, \beta_U\}$.

Chapter 9

Experimental results for MNL algorithms

In this chapter, we describe our preliminary experimental evaluation of the performance of the MNL approximation algorithm described in Chapter 7. Among various MNL problems, we select the multiple topology broadcast (MTB) problems for our experiments, since its theoretical approximation bounds are the worst, compared to the unicast and convergecast problems. For practical efficiency of the MTB approximation algorithm, we implement the MTB approximation algorithm based on an alternative LP formulation of the CWDCKOF problem, which has a polynomial number of constraints and variables (details will be given later). To investigate the quality of the MTB approximation algorithm, we compare our approximate solution, obtained from the MTB approximation algorithm, with an upper bound on the optimal solution k_{opt}. This upper bound k_{ub} is the largest integer k such that the LP-polytope $P_{LP}^{COF}(k, B : N)$ is not empty (the definition is given in Section 7.5.1). For comparison of the performance, we also implement a simple heuristic for the MTB problem.

The MTB approximation algorithm discussed in Chapter 7, uses a fixed value of parameter β, which is set to 5 as in Theorem 6.1. With this value of β, we observed that our computed solution k^* is at least $\lfloor k_{ub}/\beta \rfloor$, which implies that $\lfloor k_{opt}/\beta \rfloor$. Hence, we obtained the number of rounds k^* as expected because of Theorem 6.1. However, we discovered that the simple heuristic gave better performances in many cases. To achieve better practical performance

of the MTB approximation algorithm, we enclose it in a binary search framework, which optimises the value of β.

This chapter consists of the following parts. In Section 9.1, we discuss the implementation of the MTB approximation algorithm. In Section 9.2 we describe the simple heuristic for the MTB problem. In Section 9.3, we describe the binary search framework. In Section 9.4, we present our experimental results.

9.1 Implementation of the algorithm

In Section 7.3, we give a pseudo-polynomial time approximation algorithm for the MTB problem. In this approach, we solve the MTB problem by first finding a good value of k using binary search and the LP-relaxation of the MTB problem. With the computed k, we formulate the MTB problem as the Weighted-degree Constrained k-Outconnected subgraph (WDCKOS) problem and apply the WDCKOS approximation algorithm to compute a subgraph of multigraph G_k which satisfies the energy constraints. We then obtain the desired collection of broadcast trees by applying the Edmonds' theorem for packing arborescences. Recall that when we apply the binary search and the WDCKOS approximation algorithm, we need to solve LP-polytopes $P_{LP}^{OS}(k,B:N_k)$ and the residual LP-polytopes $P_{LP}^{OS}(k,B:J,W)$ (see Section 7.5) which have $O(km)$ variables and exponential number of constraints $O(2^n)$. Since the value of k can be very large, implementing the MTB algorithm using this approach is not practical.

In Section 7.5, we show that the MTB approximation algorithm can be implemented to run in polynomial time by formulating the MTB problem as the CWDCKOF problem (that is, the capacitated version of the WDCKOS problem) and by applying the algorithm for the CWDCKOF problem given in Figure 7.2. In the CWDCKOF algorithm, we required to compute a basic feasible solution for the initial LP polytope $P_{LP}^{COF}(k,B:N)$ and residual LP-polytopes $P_{LP}^{COF}(k,B;Y,W)$ (see Section 7.5.3 for their definitions), which have exponential

number of constraints $O(2^n)$ and m variables. Although these LP-polytopes can be solved in polynomial-time using the ellipsoid method, an additional algorithm needs to be plugged-in at each iteration of the CWDCKOF algorithm in order to transform a feasible solution, obtained from the Ellipsoid method, into a basic feasible solution. Jain [27] shows that this can be done in $O(m)P(m)+O(m^2n(L+logm))M(m,n)$ times, where L is the maximum size of the numbers involved if they are represented in binary, $P(m)$ is the time to multiply two $M \times M$ matrices, and $M(m,n)$ is time to compute one max-flow algorithm. Therefore, this approach also does not lead to a practically efficient algorithm.

Thus, we consider an alternative integer programming formulation of the CWDCKOF problem, which has a polynomial $O(nm+n^2)$ number of constraints and $nm+m+1$ variables. The alternative integer program $P^F_{IP}(B)$ can be formulated in the following way, with edge-capacity variables $y(u,v)$, for $(u,v) \in E' = \{(u,v) \in E : w(u,v) \leq B(u)\}$ and flow variables $f^x(u,v)$, for all $x \in V$ and for all $(u,v) \in E'$.

Maximise k

Subject to:

$$\sum_{v \in V} f^x(u,v) - \sum_{v \in V} f^x(v,u) = \begin{cases} k & \text{if } u = r, \\ -k & \text{if } u = s, \\ 0 & \text{else,} \end{cases} \quad \text{for all } x \in V, \text{ for all } u \in V,$$

$$f^x(u,v) \leq y(u,v), \quad \text{for all } x \in V, \text{ for all } (u,v) \in E',$$

$$\sum_{(u,v) \in E'} (y(u,v) \cdot w(u,v)) \leq B(u), \quad \text{for all } u \in V,$$

$$f^x(u,v), y(u,v) \in \mathbb{Z}, \quad \text{for all } x \in V, \text{ for all } (u,v) \in E'.$$

A fractional variant of the multiple topology broadcast problem (MTB) can be formulated by the LP-relaxation of this integer program problem [57].

Let $P^F_{LP}(B)$ denote the LP-relaxation of the integer program $P^F_{IP}(B)$, obtained by relaxing the integer constraints on the variables to $0 \leq f^x(u,v) \leq k$ and $0 \leq y(u,v) \leq k$. Observe that now k

(the number of rounds) is a variable for the LP problem $P^F_{LP}(B)$. Therefore, in this approach we do not require to find the value of k using binary search. We simply obtain the value of k by solving the initial LP $P^F_{LP}(B)$. In each iteration of the CWDCKOF algorithm, we now find a basic feasible solution to the following residual polytope$P^F_{LP}(B;Y,W,k)$, defined by the following constraints:

$$\begin{aligned}
\sum_{v\in V} f^x(u,v) - \sum_{v\in V} f^x(v,u) &= \begin{cases} \lfloor k \rfloor & \text{if} \quad u = r, \\ -\lfloor k \rfloor & \text{if} \quad u = s, \\ 0 & \text{else,} \end{cases} && \text{for all } x \in V, \text{ for all } u \in V, \\
f^x(u,v) &\leq Y(u,v) + y(u,v), && \text{for all } x \in V, \text{ for all } (u,v) \in E', \\
\sum_{(u,v)\in E'} y(u,v)\cdot w(u,v) &\leq B(v) - \sum_{e\in\delta^{out}(v)} Y(e)/\alpha, && \text{for all } u \in W, \\
0 \leq f^x(u,v), y(u,v) &\leq k, && \text{for all } x \in V, \text{ for all } (u,v) \in E'.
\end{aligned}$$

The pseudo-code of the MTB approximation algorithm is given in Figure 9.1. It can be shown similarly as in Section 7.5.3 that the algorithm of Figure 9.1 gives the solutions for the CWDCKOF problem.

The output of the MTB approximation algorithm in Figure 9.1 returns integral capacities Y which supports k^*-out-flows from r and which satisfies the energy constraints (6.1). In our experiment we are only interested in the quality of the approximate solution, i.e, the number of broadcast round that can be computed in polynomial time, and not the actual route of the individual broadcast communication. Therefore, we have not computed broadcast trees from the output capacitated graph $N = (V,E,Y)$. As we discussed in Section 7.5.2, Theorem 7.8 for packing arborescences implies that we can retrieve k^* broadcast trees in additional polynomial time.

We use Simplex method [9] as the LP algorithm as it works well in practice and it also provides a basic feasible solution without any further computation. Therefore, the running time of the algorithm of the MTB approximation algorithm is no longer polynomial.

Input: A network $N = (V, E, w, B/\beta)$ and a node $r \in V$, where $\beta = 5$
Output: integer k^* and integral capacities $Y = Y(e_1), \ldots, Y(e_m)$

Initialization: $E \leftarrow E \backslash \{(u,v) \in E : w(u,v) > B(u)\}$, $Y(e) \leftarrow 0$ for each $e \in E$,
$W \leftarrow V$, $I \leftarrow \emptyset$, $\beta = \alpha + \triangle$ where $\alpha = 2$ and $\triangle = 3$.
`/* 1st iteration */`
Remove from W every $v \in W$ with $|\delta_E^{out}(v)| \leq \triangle$.
Find a basic feasible solution $(f_{u,v}^x,\ y,\ k^*) \in P_{LP}^F(B/\beta)$.
if $k = 0$ **then**
    Return "UNFEASIBLE" and STOP.
else
    forall the $e \in E$ **do**
        $Y(e) \leftarrow \lfloor y(e) \rfloor$;
    end
    foreach $e \in E$ **do**
        if $y(e) - Y(e) \geq 1/\alpha$ **then**
            $Y(e) = Y(e) + 1$;
            Remove e from E (add them to I)
    end
    Remove from E all edges with $y(e)$ integral (add them to I)
`/* the 2nd and the subsequent iterations */`
while $E \neq \emptyset$ **do**
    Find basic solution $(f_{u,v}^x, y) \in P_{LP}^F(B/\beta; Y, W, k)$.
    Remove from E all edge with $y(e) = 0$.
    Set $Y(e) \leftarrow Y(e) + 1$ and remove e from E (add to I) for each edge e with $y(e) \geq 1/\alpha$.
    Remove from W every $v \in W$ with $|\delta_E^{out}(v)| \leq \triangle$.
end
Return Y;

Figure 9.1 The approximation algorithm for the multiple topology broadcast (MTB) problem

9.2 Simple heuristic

For the performance comparison, we have also implemented a greedy heuristic for the MTB problem, which is based on breadth-first search (BFS). This heuristic works in the following way. We first find a fractional solution to $(f_{uv}^x, y, k) \in P_{LP}^F(B)$. Now for each edge $e \in E$, we set the edge-capacity $Y(e) = \lfloor y(e) \rfloor$. Given these capacities, the heuristic performs a number of iterations. At each iteration, find a breath-first search spanning tree (broadcast tree) T rooted at the source and find the minimum capacity $c(T)$ of all edges in T, i.e, $c(T) = \min\{Y(e) : e \in T\}$.

This minimum capacity $c(T)$ is the number of times this broadcast tree T is included in our solution. At the end of the iteration, we reduce capacity $Y(e)$ by $c(T)$, for all edges $e \in T$. This iterative process is continued until there is no tree found (that is, when not all nodes are reachable from the source). A solution for MTB problem is $\sum c(T)$ rounds and the calculated collection of broadcast trees.

9.3 Optimising the parameter β

The approximation algorithm for the MTB problem, discussed in Chapter 7, uses a fixed value of parameter β, which is set to 5. With this value of β, we observed that our computed solution satisfies $k^* \geq \lfloor k_{ub}/\beta \rfloor$, which implies that $k^* \geq \lfloor k_{opt}/\beta \rfloor$. To obtain better results we optimise the value of β as follows.

Let $k(\beta)$ be the value of k (number of rounds) obtained from the MTB approximation algorithm of Theorem 6.1 when initial battery capacity $B(v)$ is set to $B(v)/\beta$ for each node $v \in V$. Let β_{opt} be the minimum β such that the MTB approximation algorithm of Theorem 6.1 returns a solution which satisfies the energy constraints B. We denote this solution by $k(\beta_{opt})$. Let $\beta_{\min} = 1$ and $\beta_{\max} = 5$. We find β_* using a binary search over the range $[\beta_{\min}, \beta_{\max}]$. In each iteration, we apply the algorithm MTB approximation algorithm of Theorem 6.1 with B set to B/β_*, where $\beta_* = \frac{\beta_{\min}+\beta_{\max}}{2}$ and check whether the obtained solution satisfies the energy constraints. If it violates the energy constraints (6.1), then we set $\beta_{\max} = \beta_*$, otherwise, we set $\beta_{\min} = \beta_*$. We stop the binary search if $\beta_{\max} - \beta_{\min}$ is relatively small. At the end of the binary search, it returns $k(\beta_*)$ solution which satisfies the energy constraints (6.1).

Following the analysis of the MTB approximation algorithm given in Section 7.3.2, we show that $k(\beta_*) \geq \lfloor k_{opt}/\beta_* \rfloor$, for $\beta_0 \geq \beta_* \geq \beta_{opt}$, where $\beta_0 = 5$. By Lemma 7.3 and 7.4, we know that $P_{LP}^{OS}(\lfloor k_{opt}/\beta_* \rfloor, B/\beta_* : N_K)$ is not empty for any $\beta_* \geq 1$. Suppose that the binary search returns some value of β_*. The binary search always returns $k(\beta_*)$ solution, which satisfies the energy constraints (this follows from the algorithm). Hence, Lemma 7.1 implies that LP-polytope $P_{LP}^{OS}(k(\beta_*), B/\beta_* : N_K)$ is not empty. Thus, $k(\beta_*) \geq \lfloor k_{opt}/\beta_* \rfloor$.

9.4 Experimental results

In this section, we evaluate the quality of the approximate solution for the multiple topology broadcast (MTB) approximation algorithm presented in Chapter 7. We compare our approximate solution with the simple heuristic presented in this chapter. Recall that the optimal number of rounds (k_{opt}) for the MTB problem is the largest integer k such that the integer program $P_{LP}^{COF}(k, B:N)$ is feasible. Due to the time complexity of finding k_{opt}, we compare our approximate solution to an upper bound on the optimal solution. This upper bound k_{ub} is the largest integer k such that LP-relaxation of the IP $P_{LP}^{COF}(k, B:N)$ is not empty. We obtain this upper bound by simply solving the initial LP problem $(f_{(u,v)}^{x}, y, k_{ub}) \in P_{LP}^{F}(B)$.

9.4.1 Development and experiments platform

The experiments were run on a Windows 7 64-bit Operating System, with Intel(R) Core(TM) i7-2600 CPU @ 3.40 GHz having installed RAM of 8.00 GB. We implemented the MTB approximation algorithm and the heuristic using Matlab. For the linear programming algorithm, we used the Simplex method LINPROG, which is included in Matlab's optimization toolbox. We have also used MATLAB Bioinformatics toolbox for the graph-related algorithms, such as depth-first search.

9.4.2 Input instances

For each network topology, the network sizes n are varied to be 20, 30, 40, and 50, respectively. The nodes are randomly generated and uniformly distributed in a 100×100 grid. With the generated nodes, we first create a complete graph. Then, for each node, we choose h closest nodes to be its 1-hop neighbours, i.e, an edge $(u, v) \in E$ exists if node v is one of the h closest nodes from node u. All other edges are discarded. In this configuration, the degree of node is set to h and it is the same for all nodes. Therefore, the number of edges in the graph is hn.

We experiment with $h = 5$ and 10. The source node is located at the center of the field, i.e, its coordinate is $(50,50)$. An edge-weight $w(u,v)$, which represents a transmission cost of sending a message from node u to node v, is set to $d(u,v)^{\gamma}$, where $d(u,v)$ is the Euclidean distance from node u to node v. We set the propagation loss exponents $\gamma = 2$.

For each node u the initial battery capacity B is set to $\frac{\sum w(u,v)}{deg(u)} \times 10$. The reason behind this setting is to force that each node has enough initial battery capacity to support on average of 10 transmissions before it depletes all its energy.

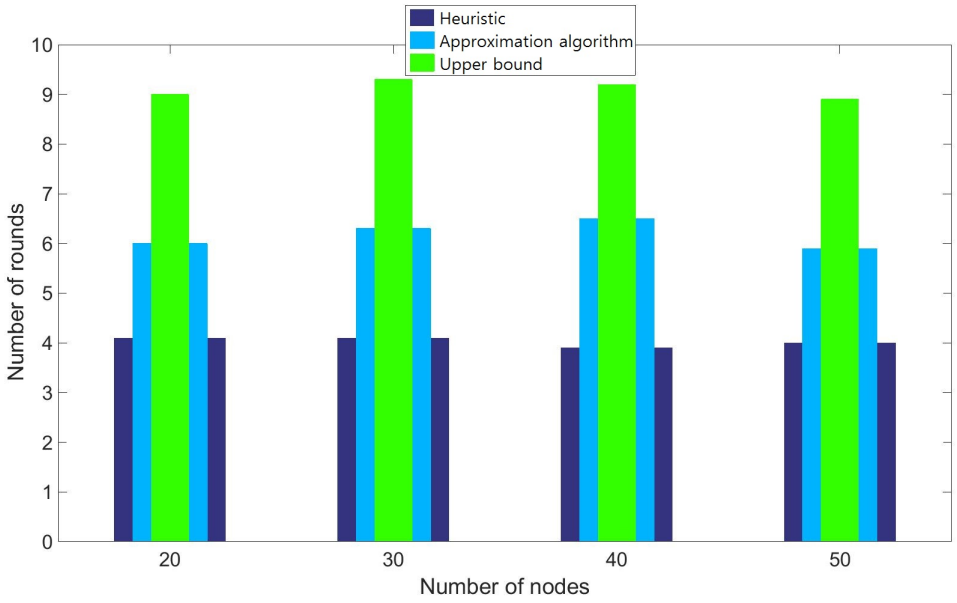
9.4.3 Results

Figures 9.2(a) and 9.2(b) show the comparison of the results obtained from the MTB approximation algorithm (with optimised β), the heuristic and the upper bounds on the optimal solution. In these figures, the x-axis indicates the computed number of rounds while the y-axis indicates the number of nodes in the graph. For each network size n, we generate 10 random instances and run for each instance. Each plot is obtained by averaging the 10 runs. The light green bar represents the computed upper bound of the optimal solution, the light blue bar represents the results obtained from the MTB approximation algorithm, and the dark blue bar represents the results obtained from the heuristic.

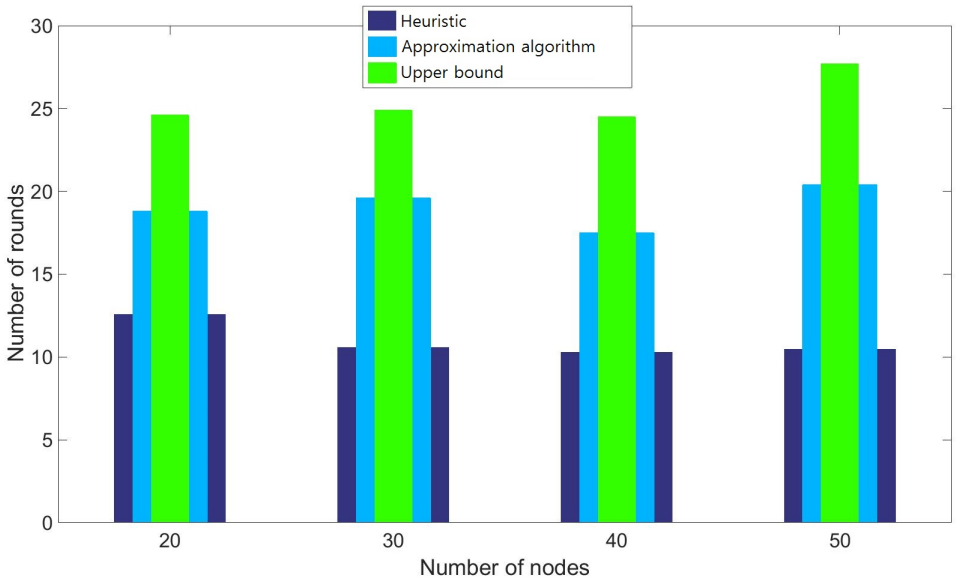
Figure 9.2(a) shows the results when each node has degree of 5 and Figure 9.2(b) shows the results when each node has degree of 10. From these figures we can observe that the MTB approximation algorithm always gives better performance than the heuristic. In Figure 9.2(a), the MTB approximation algorithm computes on average (approximately) one and two thirds times more rounds than the heuristic. This difference becomes even greater when degrees of a node is set to 10 (see Figure 9.2(b)).

Tables 9.1 and 9.2 show the value of β_* used when we obtain the results for the MTB approximation algorithm, which are shown in Figure 9.2(a) and 9.2(b), respectively. We showed in Section 9.3 that $\lfloor k_{opt}/\beta_* \rfloor \leq k$. From Tables 9.1 and 9.2 we can observe that interestingly

the value of β_* is never greater than 2. This may be evident that the approximation bounds for the MTB approximation algorithm given in Theorem 6.1 may not be tight and could be further improved.



(a) Each node has degree of 5



(b) Each node has degree of 10

Figure 9.2 Performance comparisons: The MTB approximation algorithm, the heuristic, and the upper bound

	network size n			
Instance	20	30	40	50
1	1.44	1.49	1.20	1.54
2	1.52	1.66	1.51	1.53
3	1.49	1.32	1.45	1.43
4	1.36	1.32	1.54	1.54
5	1.56	1.38	1.38	1.44
6	1.15	1.31	1.34	1.56
7	1.33	1.21	1.49	1.49
8	1.49	1.51	1.58	1.39
9	1.33	1.47	1.41	1.53
10	1.25	1.52	1.22	1.52

Table 9.1 The value of β_* used when we obtain the results of the MTB approximation algorithm shown in Figure 9.2(a)

9.4.4 Discussion

In these preliminary experiments, we focused on evaluating the quality of the computed solution. We did not measure the time taken by the MTB approximation algorithm and the heuristic. However, we give some indication about the execution time. Throughout the experiments we observed that the naive MTB approximation algorithm (without optimising β) and the heuristic have similar execution times even though MTB approximation algorithm needs to iteratively solve the LP problems. We observe that this is because after the first iteration of the MTB approximation algorithm, most of edges are removed from the graph so that the residual LP problem has relatively small number of variable and constraints. The MTB approximation algorithm usually terminates within three iterations. The bottleneck of both MTB approximation algorithm and the heuristic is the solving the initial LP problem. The execution times of the MTB approximation algorithms with the optimisation β are much slower than the heuristic. This is because in each iteration of binary search we need to apply the entire MTB approximation algorithm and check whether the obtained solutions violates the energy constraints.

Our results are preliminary hence it requires further developments. The simple heuristic needs to solve the LP-relaxation of the original problem in order to find the value of k. Therefore

	network size n			
Instance	20	30	40	50
1	1.24	1.23	1.23	1.19
2	1.66	1.66	1.75	1.47
3	1.58	1.24	1.25	1.32
4	1.30	1.18	1.80	1.38
5	1.15	1.26	1.20	1.22
6	1.20	1.86	1.24	1.23
7	1.23	1.24	1.97	1.97
8	1.26	1.75	1.24	1.36
9	1.10	1.44	1.68	1.23
10	1.75	1.22	1.31	1.25

Table 9.2 The value of β_* used when we obtain the results of the MTB approximation algorithm shown in Figure 9.2(b).

as the network size grows, the heuristic becomes impractical. Hence, further developments of good heuristics without LP problems are required. We would also need to create other input instances and conduct rigorous experiments.

Chapter 10

Conclusion

In this thesis, we considered the Directed Weighted Degree Constrained Network Design (DWDCN) problems and their applications to the Maximum Network Lifetime (MNL) problems in wireless ad-hoc networks.

The DWDCN problems have many variants depending on the type of the connectivity requirements and on the type of the degree bounds. We considered a general case when the connectivity requirements were defined by an arbitrary intersecting or crossing supermodular set function and the degree bounds were defined for the out-degrees of nodes or the in-degrees or both. We followed the approach proposed by Nutov [17, 18], who developed polynomial time bi-criteria approximation algorithms for these problems. By developing more detailed analysis of the approximation algorithms we obtained better approximation bounds for many DWDCN problems.

Since the DWDCN problems are defined for general connectivity requirements, algorithms for these problems can be applied in various types of network design problems with weighted degree constraints, including finding a Weighted Degree Constrained k-Outconnected subgraph problem and Weighted Degree Constrained Out-Arborescence problem. We applied DWDCN approximation algorithms to a class of Maximum Network Lifetime (MNL) problems in

wireless ad-hoc networks. We considered MNL broadcast, convergecast, unicast and mixedcast problems. Using our new approximation bounds for the DWDCN problems, we improved previous approximation bounds for the MNL problems.

We also conducted experimental evaluation of the multiple topology broadcast (MTB) approximation algorithm. We observed that the computed solutions are always better than our theoretical lower bounds, as expected. To investigate the quality of the MTB approximation algorithm, we compared our approximate solutions to solutions obtained by heuristic. We discovered that the simple heuristic gave better performances in many cases. To achieve better performance of the MTB approximation algorithm, we optimised the approximation algorithm by parameterizing the value β, where $\beta = 5$ is a constant. By doing so, we observed a clear improvement. The MTB approximation algorithm always gave the better performance than the heuristic.

We conclude the thesis by providing some potential directions for future research.

*Directed Weighted Degree Constrained Network Design (*DWDCN*) problems*

One natural question is whether the approximation bounds for the DWDCN problems given in this thesis can be further improved using our methods. Bansal *et al.* gave additive (plus 4) approximation algorithm for the DWDCN problem with unit-weights and intersecting supermodular set function f. In order to obtain such additive guarantees on the degree bounds, the cost of the subgraph becomes unbounded. By exploiting this cost-degree trade-off and adapting it into the DWDCN problem with weighted degree constraints, we may improve the approximations on the weighted degree bounds, which will result in improving approximation bounds for the MNL problems.

We considered the DWDCN problems with intersecting and crossing supermodular set function f. This set function f can be used to define various connectivity requirements, including k-edge-outconnected with root r, out- or in-arborescence rooted at r, and strongly k-edge-connected. However, there exist connectivity requirements that cannot be defined with

such set functions. For example, the connectivity requirement for the directed Steiner Tree problem or the directed Steiner network problem. Such connectivity requirements can be defined by a weakly supermodular function. To the best of our knowledge, we did not find any literature related to DWDCN problems with weakly supermodular functions. Therefore, developing an algorithm for the DWDCN with weakly supermodular function may be a possible research direction.

In this thesis, we focused on the edge-connectivity type of the network design problems. Recently degree-bounded network design problems with *node-connectivity* requirements [18, 14] have received much attention. Therefore developing an algorithm for these types of network design problem will be an interesting research topic.

*Maximum Network Lifetime (*MNL*) problems*

Our proofs of the approximation bounds of the multiple topology MNL algorithms given in Chapter 7 hold only for the input instances which do not contain edges (u,v) with $w(u,v) > B(u)/\beta$, where β is the constant given in Theorem 6.1. This condition is critical in the analysis of approximation bounds of the DWDCN algorithms, but is not a natural part of the definition of the problem. Developing approximation algorithms for the MNL problems without having such condition would be an interesting research topic.

Another possible direction of research is finding more practical algorithms. The MNL approximation algorithm discussed in this thesis is based on iterative rounding (relaxation) method. Hence, at each iteration, a basic solution of an LP relaxation, which has exponential number of constraints, needs to be computed. Although as discussed in Chapter 9, we can use an alternative formulation, which has polynomial number of constraints and variables, it still needs high computational costs for larger networks. Hence development of practical algorithms or good heuristic for the MNL problems is worth investigation. We have considered "centralised" approach in which a priori knowledge of the full network topology is assumed. However this may not be always possible in certain scenarios. Hence developing distributed algorithm for MNL problem is also an interesting research direction.

Bibliography

[1] Nikhil Bansal, Rohit Khandekar, and Viswanath Nagarajan. Additive guarantees for degree bounded directed network design. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 769–778, 2008.

[2] Nikhil Bansal, Rohit Khandekar, and Viswanath Nagarajan. Additive guarantees for degree-bounded directed network design. *SIAM J. Comput.*, 39(4):1413–1431, 2009.

[3] Fred Bauer and Anujan Varma. Degree-constrained multicasting in point-to-point networks. In *Proceedings IEEE INFOCOM '95, The Conference on Computer Communications, Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Bringing Information to People, Boston, Massachusetts, USA, April 2-6, 1995*, pages 369–376, 1995.

[4] Anand Bhalgat, Ramesh Hariharan, Telikepalli Kavitha, and Debmalya Panigrahi. Fast edge splitting and edmonds' arborescence construction for unweighted graphs. In *SODA*, pages 455–464, 2008.

[5] Fang Bian, Ashish Goel, Cauligi S. Raghavendra, and Xin Li. Energy-efficient broadcasting in wireless ad hoc networks lower bounds and algorithms. *Journal of Interconnection Networks*, 3(3-4):149–166, 2002.

[6] Hans L. Bodlaender, Richard B. Tan, Thomas C. van Dijk, and Jan van Leeuwen. Integer maximum flow in wireless sensor networks with energy constraint. In *Algorithm Theory - SWAT 2008, 11th Scandinavian Workshop on Algorithm Theory, Gothenburg, Sweden, July 2-4, 2008, Proceedings*, pages 102–113, 2008.

[7] Imane Horiya Brahmi, Soufiene Djahel, Damien Magoni, and John Murphy. An efficient partial data aggregation scheme in wsns. In *2014 IFIP Wireless Days, WD 2014, Rio de Janeiro, Brazil, November 12-14, 2014*, pages 1–3, 2014.

[8] Andrea E. F. Clementi, Pierluigi Crescenzi, Paolo Penna, Gianluca Rossi, and Paola Vocca. On the complexity of computing minimum energy consumption broadcast subgraphs. In *STACS 2001, 18th Annual Symposium on Theoretical Aspects of Computer Science, Dresden, Germany, February 15-17, 2001, Proceedings*, pages 121–131, 2001.

[9] G. B. Dantzig. *Maximization of a Linear Function of Variables Subject to Linear Inequalities, in Activity Analysis of Production and Allocation*, chapter XXI. Wiley, New York, 1951.

[10] Koustuv Dasgupta, Konstantinos Kalpakis, and Parag Namjoshi. An efficient clustering-based heuristic for data gathering and aggregation in sensor networks. In *2003 IEEE Wireless Communications and Networking, WCNC 2003, New Orleans, LA, USA, 16-20 March, 2003*, pages 1948–1953, 2003.

[11] Guofeng Deng and Sandeep K. S. Gupta. Maximizing broadcast tree lifetime in wireless ad hoc networks. In *GLOBECOM*, 2006.

[12] J. Edmonds. Edge-disjoint branchings. In B. Rustin, editor, *Combinatorial Algorithms*, pages 91–96. Academic Press, 1973.

[13] Michael Elkin, Yuval Lando, Zeev Nutov, Michael Segal, and Hanan Shpungin. Novel algorithms for the network lifetime problem in wireless settings. *Wireless Networks*, 17 (2):397–410, 2011.

[14] Alina Ene and Ali Vakilian. Improved approximation algorithms for degree-bounded network design problems with node connectivity requirements. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 754–763, 2014.

[15] Thomas Erlebach, Tom Grant, and Frank Kammer. Maximising lifetime for fault-tolerant target coverage in sensor networks. In *SPAA 2011: Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, San Jose, CA, USA, June 4-6, 2011 (Co-located with FCRC 2011)*, pages 187–196, 2011.

[16] András Frank. Kernel systems of directed graphs. *Acta Sci Math (Szeged)*, 41:63–76, 1979.

[17] Takuro Fukunaga and Hiroshi Nagamochi. Network design with weighted degree constraints. In *WALCOM: Algorithms and Computation, Third International Workshop, WALCOM 2009, Kolkata, India, February 18-20, 2009. Proceedings*, pages 214–225, 2009.

[18] Takuro Fukunaga, Zeev Nutov, and R. Ravi. Iterative rounding approximation algorithms for degree-bounded node-connectivity network design. *SIAM J. Comput.*, 44(5):1202–1229, 2015.

[19] Harold N. Gabow and K. S. Manu. Packing algorithms for arborescences (and spanning trees) in capacitated graphs. *Math. Program.*, 82:83–109, 1998.

[20] Shashidhar Gandham, Milind Dawande, Ravi Prakash, and Subbarayan Venkatesan. Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In *Proceedings of the Global Telecommunications Conference, 2003. GLOBECOM '03, San Francisco, CA, USA, 1-5 December 2003*, pages 377–381, 2003.

[21] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. ISBN 0716710447.

[22] Ashish Goel and Deborah Estrin. Simultaneous optimization for concave costs: single sink aggregation or single source buy-at-bulk. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.*, pages 499–505, 2003.

[23] Michel X. Goemans. Minimum bounded degree spanning trees. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 273–282, 2006.

[24] Javier Gomez and Andrew T. Campbell. A case for variable-range transmission power control in wireless multihop networks. In *INFOCOM*, 2004.

[25] Clóvis C Gonzaga. An algorithm for solving linear programming problems in o (n 3 l) operations. In *Progress in mathematical programming*, pages 1–28. Springer, 1989.

[26] Yiwei Thomas Hou, Yi Shi, and Hanif D. Sherali. On node lifetime problem for energy-constrained wireless sensor networks. *MONET*, 10(6):865–878, 2005.

[27] Kamal Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[28] Konstantinos Kalpakis and Shilang Tang. A combinatorial algorithm for the maximum lifetime data gathering with aggregation problem in sensor networks. *Computer Communications*, 32(15):1655–1665, 2009.

[29] Konstantinos Kalpakis, Koustuv Dasgupta, and Parag Namjoshi. Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Computer Networks*, 42(6):697–716, 2003.

[30] Intae Kang and Radha Poovendran. Maximizing network lifetime of broadcasting over wireless stationary ad hoc networks. *MONET*, 10(6):879–896, 2005.

[31] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, pages 302–311. ACM, 1984. ISBN 0-89791-133-4.

[32] L.G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53 – 72, 1980. ISSN 0041-5553.

[33] Maleq Khan and Gopal Pandurangan. A fast distributed approximation algorithm for minimum spanning trees. In *Distributed Computing, 20th International Symposium, DISC 2006, Stockholm, Sweden, September 18-20, 2006, Proceedings*, pages 355–369, 2006.

[34] Rohit Khandekar, Guy Kortsarz, and Zeev Nutov. On some network design problems with degree constraints. *J. Comput. Syst. Sci.*, 79(5):725–736, 2013.

[35] Lap Chi Lau and Mohit Singh. Additive approximation for bounded degree survivable network design. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 759–768, 2008.

[36] Lap Chi Lau and Mohit Singh. Additive approximation for bounded degree survivable network design. *SIAM J. Comput.*, 42(6):2217–2242, 2013.

[37] Lap Chi Lau and Hong Zhou. A unified algorithm for degree bounded survivable network design. In *Integer Programming and Combinatorial Optimization - 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings*, pages 369–380, 2014.

[38] Lap Chi Lau, Joseph Naor, Mohammad R. Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 651–660, 2007.

[39] Lap Chi Lau, Joseph Naor, Mohammad R. Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints. *SIAM J. Comput.*, 39(3):1062–1087, 2009.

[40] Sang-Hyuk Lee and Tomasz Radzik. Improved approximation bounds for maximum lifetime problems in wireless ad-hoc network. In *Ad-hoc, Mobile, and Wireless Networks - 11th International Conference, ADHOC-NOW 2012, Belgrade, Serbia, July 9-11, 2012. Proceedings*, pages 14–27, 2012.

[41] Sang-Hyuk Lee and Tomasz Radzik. Approximation bounds on the number of mixedcast rounds in wireless ad-hoc networks. In *Combinatorial Algorithms - 24th International Workshop, IWOCA 2013, Rouen, France, July 10-12, 2013, Revised Selected Papers*, pages 283–296, 2013.

[42] Wuyungerile Li, Masaki Bandai, and Takashi Watanabe. Tradeoffs among delay, energy and accuracy of partial data aggregation in wireless sensor networks. In *24th IEEE International Conference on Advanced Information Networking and Applications, AINA 2010, Perth, Australia, 20-13 April 2010*, pages 917–924, 2010.

[43] Weifa Liang. Constructing minimum-energy broadcast trees in wireless ad hoc networks. In *Proceedings of the 3rd ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2002, June 9-11, 2002, Lausanne, Switzerland*, pages 112–122, 2002.

[44] Weifa Liang and Yuzhen Liu. Online data gathering for maximizing network lifetime in sensor networks. *IEEE Trans. Mob. Comput.*, 6(1):2–11, 2007.

[45] Hwa-Chun Lin, Feng-Ju Li, and Kai-Yang Wang. Constructing maximum-lifetime data gathering trees in sensor networks with data aggregation. In *ICC*, pages 1–6, 2010.

[46] Anand Louis and Nisheeth K. Vishnoi. Improved algorithm for degree bounded survivable network design problem. In *Algorithm Theory - SWAT 2010, 12th Scandinavian Symposium and Workshops on Algorithm Theory, Bergen, Norway, June 21-23, 2010. Proceedings*, pages 408–419, 2010.

[47] Ivana Maric and Roy D. Yates. Cooperative multicast for maximum network lifetime. *IEEE Journal on Selected Areas in Communications*, 23(1):127–135, 2005.

[48] Vardges Melkonian and Éva Tardos. Approximation algorithms for a directed network design problem. In *Integer Programming and Combinatorial Optimization, 7th International IPCO Conference, Graz, Austria, June 9-11, 1999, Proceedings*, pages 345–360, 1999.

[49] Vardges Melkonian and Éva Tardos. Algorithms for a network design problem with crossing supermodular demands. *Networks*, 43(4):256–265, 2004.

[50] Zeev Nutov. Approximating directed weighted-degree constrained networks. In *APPROX-RANDOM*, pages 219–232, 2008.

[51] Zeev Nutov. Approximating maximum integral flows in wireless sensor networks via weighted-degree constrained k-flows. In *Proceedings of the DIALM-POMC Joint Workshop on Foundations of Mobile Computing, Toronto, Canada, August 18-21, 2008*, pages 29–34, 2008.

[52] Zeev Nutov. Approximating directed weighted-degree constrained networks. *Theor. Comput. Sci.*, 412(8-10):901–912, 2011.

[53] Zeev Nutov. Degree constrained node-connectivity problems. *Algorithmica*, 70(2): 340–364, 2014.

[54] Zeev Nutov and Michael Segal. Improved approximation algorithms for maximum lifetime problems in wireless networks. In *ALGOSENSORS*, pages 41–51, 2009.

[55] Zeev Nutov and Michael Segal. Improved approximation algorithms for maximum lifetime problems in wireless networks. *Theor. Comput. Sci.*, 453:88–97, 2012.

[56] Carlos A. S. Oliveira and Panos M. Pardalos. A survey of combinatorial optimization problems in multicast routing. *Comput. Oper. Res.*, 32(8):1953–1981, August 2005. ISSN 0305-0548.

[57] Ariel Orda and Ben-Ami Yassour. Maximum-lifetime routing algorithms for networks with omnidirectional and directional antennas. In *MobiHoc*, pages 426–437, 2005.

[58] J. Park and Salim Sahni. Maximum lifetime broadcasting in wireless networks. In *AICCSA*, page 8, 2005.

[59] R. Ravi, Madhav V. Marathe, S. S. Ravi, Daniel J. Rosenkrantz, and Harry B. Hunt III. Approximation algorithms for degree-constrained minimum-cost network-design problems. *Algorithmica*, 31(1):58–78, 2001.

[60] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Comb. Theory, Ser. B*, 80(2):346–355, 2000.

[61] Michael Segal. Fast algorithm for multicast and data gathering in wireless networks. *Inf. Process. Lett.*, 107(1):29–33, 2008.

[62] Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 661–670, 2007.

[63] Chao Song, Ming Liu, Jiannong Cao, Yuan Zheng, Hai gang Gong, and Guihai Chen. Maximizing network lifetime based on transmission range adjustment in wireless sensor networks. *Computer Communications*, 32(11):1316–1325, 2009.

[64] Jeffrey Stanford and Sutep Tongngam. Approximation algorithm for maximum lifetime in wireless sensor networks with data aggregation. In *SNPD*, pages 273–277, 2006.

[65] Hüseyin Özgür Tan and Ibrahim Korpeoglu. Power efficient data gathering and aggregation in wireless sensor networks. *SIGMOD Record*, 32(4):66–71, 2003.

[66] Niwat Thepvilojanapong, Yoshito Tobe, and Kaoru Sezaki. On the construction of efficient data gathering tree in wireless sensor networks. In *International Symposium on Circuits and Systems (ISCAS 2005), 23-26 May 2005, Kobe, Japan*, pages 648–651, 2005.

[67] Pascal von Rickenbach, Roger Wattenhofer, and Aaron Zollinger. Algorithmic models of interference in wireless ad hoc and sensor networks. *IEEE/ACM Trans. Netw.*, 17(1): 172–185, 2009.

[68] Peng-Jun Wan, Xiang-Yang Li, and Ophir Frieder. Minimum energy cost broadcasting in wireless networks. In *Encyclopedia of Algorithms*. 2008.

[69] Jeffrey E. Wieselthier, Gam D. Nguyen, and Anthony Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *INFOCOM*, pages 585–594, 2000.

[70] Yan Wu, Sonia Fahmy, and Ness B. Shroff. On the construction of a maximum-lifetime data gathering tree in sensor networks: Np-completeness and approximation algorithm. In *INFOCOM*, pages 356–360, 2008.

[71] Mario Zagalj, Jean-Pierre Hubaux, and Christian C. Enz. Minimum-energy broadcast in all-wireless networks: : Np-completeness and distribution issues. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking, MOBICOM 2002, Atlanta, Georgia, USA, September 23-28, 2002*, pages 172–182, 2002.

[72] Yihua Zhu, Wan-deng Wu, Jian Pan, and Yi-ping Tang. An energy-efficient data gathering algorithm to prolong lifetime of wireless sensor networks. *Computer Communications*, 33(5):639–647, 2010.